

## PATENT ABSTRACTS OF JAPAN

(11) Publication number: **10320205 A**(43) Date of publication of application: **04.12.98**

(51) Int. Cl.

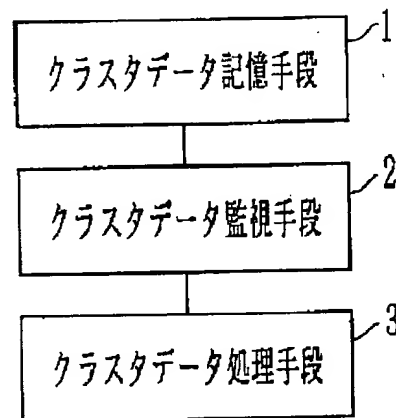
**G06F 9/44****G06F 9/44**(21) Application number: **09130160**(22) Date of filing: **20.05.97**(71) Applicant: **FUJITSU LTD**(72) Inventor: **NISHIGAYA TAKESHI  
IIDA ICHIRO**(54) **INFORMATION PROCESSOR**

## (57) Abstract:

PROBLEM TO BE SOLVED: To delete or substitute a part of a program code without stopping a program in operating.

SOLUTION: A class data monitoring means 2 monitors the using state of class data stored in a class data storing means 1 in each added class data and a class data processing means 3 deletes or substitutes a part of a program during the operation of the program by deleting or substituting class data based on the using state of the class data monitored by the means 2.

COPYRIGHT: (C)1998,JPO



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-320205

(43) 公開日 平成10年(1998)12月4日

(51) Int.Cl.<sup>6</sup>

G 0 6 F 9/44

識別記号

5 3 0

5 5 2

F I

G 0 6 F 9/44

5 3 0 M

5 5 2

審査請求 未請求 請求項の数44 O L (全 43 頁)

(21) 出願番号

特願平9-130160

(22) 出願日

平成9年(1997)5月20日

(71) 出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番  
1号

(72) 発明者 西ヶ谷 岳

神奈川県川崎市中原区上小田中4丁目1番  
1号 富士通株式会社内

(72) 発明者 飯田 一朗

神奈川県川崎市中原区上小田中4丁目1番  
1号 富士通株式会社内

(74) 代理人 弁理士 大菅 義之 (外1名)

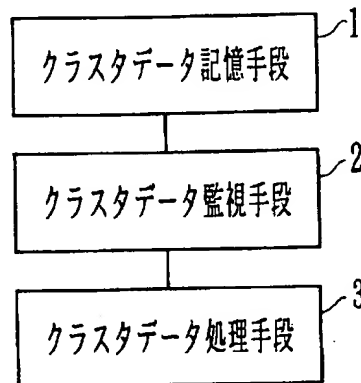
(54) 【発明の名称】 情報処理装置

(57) 【要約】

【課題】 動作中のプログラムを停止させることなく、そのプログラムコードの一部を削除したり、置き換えたりすることを可能とする。

【解決手段】 クラスデータ監視手段2は、クラスデータ記憶手段1に記憶されているクラスデータの使用状況を追加されたクラスデータごとに監視し、クラスデータ処理手段3は、クラスデータ監視手段2で監視されているクラスデータの使用状況に基づいて、クラスデータの削除や置換を行うことにより、プログラムの動作中に、そのプログラムの一部の削除や置換を行えるようにする。

本発明の一実施例に係わる  
情報処理装置の機能的な構成を示すブロック図



## 【特許請求の範囲】

【請求項1】 オブジェクトに追加されたクラスデータを記憶するクラスデータ記憶手段と、  
前記クラスデータの使用状況を監視するクラスデータ監視手段と、  
前記クラスデータの使用状況に基づいて、前記クラスデータの処理を行うクラスデータ処理手段とを備えることを特徴とする情報処理装置。

【請求項2】 前記クラスデータ処理手段は、  
前記クラスデータについての処理要求を受け付ける受付手段と、  
前記処理要求のあったクラスデータが使用中かどうかを判定する判定手段と、  
前記クラスデータが使用中の場合、前記クラスデータの使用が終了するまで、前記クラスデータの処理を延期する延期手段とを備えることを特徴とする請求項1に記載の情報処理装置。

【請求項3】 前記クラスデータ処理手段は、  
前記クラスデータについての処理要求を受け付ける受付手段と、  
前記処理要求のあったクラスデータを使用しているオブジェクトに中止要求を発行する中止要求発行手段と、  
前記オブジェクトが前記中止要求に応答した後、前記クラスデータの処理を行う実行手段とを備えることを特徴とする請求項1または2に記載の情報処理装置。

【請求項4】 前記受付手段は、前記クラスデータについての処理要求を受け付けるかどうかを検査する検査手段をさらに備えることを特徴とする請求項2または3に記載の情報処理装置。

【請求項5】 前記クラスデータ処理手段は、前記クラスデータの追加要求単位で前記クラスデータを前記クラスデータ記憶手段に追加するクラスデータ追加手段を備えることを特徴とする請求項1～4のいずれか1項に記載の情報処理装置。

【請求項6】 前記クラスデータ処理手段は、削除対象のクラスデータの使用が終了してから、前記クラスデータを削除するクラスデータ削除手段を備えることを特徴とする請求項1～5のいずれか1項に記載の情報処理装置。

【請求項7】 前記クラスデータ処理手段は、置換対象のクラスデータの使用が終了してから、前記クラスデータを置換するクラスデータ置換手段を備えることを特徴とする請求項1～6のいずれか1項に記載の情報処理装置。

【請求項8】 クラスデータの使用状況を監視するエージェントクラスオブジェクトを生成するためのエージェントクラスを格納するエージェントクラス格納手段と、  
前記クラスデータに対する処理が行われる場合、前記エージェントクラスに基づいて前記エージェントクラスオブジェクトを生成するエージェントクラスオブジェクト

生成手段とを備えることを特徴とする情報処理装置。

【請求項9】 前記エージェントクラスは、  
前記エージェントクラスオブジェクトに追加されたクラスデータの管理を行うクラス管理オブジェクトと、  
前記クラス管理オブジェクトで管理されるクラスデータを記憶する共有クラスデータテーブルと、  
前記クラスデータに対する処理を実行する外部公開メソッドとを備えることを特徴とする請求項8に記載の情報処理装置。

【請求項10】 前記エージェントクラスは、  
前記クラスデータの追加要求に対して生成されたクラス管理オブジェクトと前記クラスデータの追加要求時に与えられた名前との対応関係を記憶したクラス管理オブジェクトテーブルをさらに備えることを特徴とする請求項9に記載の情報処理装置。

【請求項11】 前記クラス管理オブジェクトは、  
各クラス管理オブジェクトに固有のクラスデータを格納する局所クラスデータテーブルと、  
前記クラスデータの削除要求を記憶する削除フラグと、  
前記クラスデータの置換要求時の新たなクラスデータを記憶する置換クラスデータ記憶領域と、  
前記クラスデータを実行しているスレッドの識別情報を記憶する実行中スレッドリストと、  
前記クラスデータに対応するクラス名を記憶するクラス名リストと、  
前記クラスデータの参照先のクラスデータを管理しているクラス管理オブジェクトの識別情報を記憶する参照オブジェクトリストと、  
前記クラスデータの参照元のクラスデータを管理しているクラス管理オブジェクトの識別情報を記憶する被参照オブジェクトリストとをさらに備えることを特徴とする請求項10に記載の情報処理装置。

【請求項12】 前記エージェントクラスは、他のエージェントクラスオブジェクトからのアクセス権をチェックするアクセス制御クラスを前記共有クラスデータテーブルに備えることを特徴とする請求項9～11のいずれか1項に記載の情報処理装置。

【請求項13】 前記エージェントクラスオブジェクトは、  
他のエージェントクラスオブジェクトからのアクセスがあった場合、前記アクセス制御クラスに基づいてアクセス制御オブジェクトを生成するアクセス制御オブジェクト生成手段と、  
前記アクセス制御オブジェクトに基づいて、前記アクセス権のチェックを実行するアクセス制御オブジェクト実行手段とを備えることを特徴とする請求項12に記載の情報処理装置。

【請求項14】 前記外部公開メソッドは、前記エージェントクラスオブジェクトにクラスデータを追加するクラスデータ追加手段を備えることを特徴とする請求項1

1に記載の情報処理装置。

【請求項15】 前記クラスデータ追加手段は、前記エージェントクラスオブジェクトに追加されたクラスデータを管理するクラス管理オブジェクトを生成するクラス管理オブジェクト生成手段と、前記クラス管理オブジェクト内のクラス名リストに前記クラスデータのクラス名を登録するクラス名登録手段と、前記クラスデータを前記共有クラスデータテーブルに書き込むクラスデータ書き込み手段と、前記クラスデータの追加要求時に与えられた名前を前記クラス管理オブジェクトテーブルに登録するクラス管理オブジェクト登録手段とを備えることを特徴とする請求項14に記載の情報処理装置。

【請求項16】 前記クラスデータ追加手段は、前記エージェントクラスオブジェクトに追加されたクラスデータのクラス名が、前記共有クラスデータテーブルに登録されているクラスデータのクラス名と重複する場合、前記エージェントクラスオブジェクトに追加されたクラスデータを前記共有クラスデータテーブルから削除することにより、前記共有クラスデータテーブルの元の状態を復元する共有クラスデータテーブル復元手段を備えることを特徴とする請求項15に記載の情報処理装置。

【請求項17】 前記外部公開メソッドは、前記エージェントクラスオブジェクトに保有されているクラスデータを実行するクラスデータ実行手段を備えることを特徴とする請求項13～16のいずれか1項に記載の情報処理装置。

【請求項18】 前記クラスデータ実行手段は、実行中のクラスデータを管理しているクラス管理オブジェクトの実行中スレッドリストに、前記クラスデータを実行しているスレッドの識別情報を設定する第1スレッドリスト設定手段と、実行中のクラスデータの参照先のクラスデータを管理しているクラス管理オブジェクトの実行中スレッドリストに、前記クラスデータを実行しているスレッドの識別情報を設定する第2スレッドリスト設定手段とを備えることを特徴とする請求項17に記載の情報処理装置。

【請求項19】 前記クラスデータ実行手段は、クラス管理オブジェクトの実行中スレッドリストにスレッドの識別情報が設定されているかどうかを検索する第1実行中スレッドリスト検索手段と、前記クラス管理オブジェクトの削除フラグの設定状態を判定する削除フラグ判定手段と、前記実行中スレッドリストが空で、かつ、前記削除フラグがオンのクラス管理オブジェクトが存在している場合、前記クラス管理オブジェクトのクラス名リストのクラス名に対応するクラスデータを前記共有クラスデータテーブルから削除する削除実行手段とを備えることを特

徴とする請求項18に記載の情報処理装置。

【請求項20】 前記クラスデータ実行手段は、クラス管理オブジェクトの実行中スレッドリストにスレッドの識別情報が設定されているかどうかを検索する第2実行中スレッドリスト検索手段と、前記クラス管理オブジェクトの置換クラスデータ記憶領域を検索する置換クラスデータ検索手段と、前記実行中スレッドリストが空で、かつ、前記置換クラスデータ記憶領域にクラスデータが記憶されているクラス管理オブジェクトが存在している場合、前記共有クラスデータテーブルに記憶されている置換対象のクラスデータを、前記置換クラスデータ記憶領域に記憶されているクラスデータで置換する置換実行手段とを備えることを特徴とする請求項18または19に記載の情報処理装置。

【請求項21】 前記外部公開メソッドは、前記エージェントクラスオブジェクトからクラスデータを削除するクラスデータ削除手段を備えることを特徴とする請求項13～20のいずれか1項に記載の情報処理装置。

【請求項22】 前記クラスデータ削除手段は、削除対象のクラスデータを管理しているクラス管理オブジェクトの実行中スレッドリストを検索する第3実行中スレッドリスト検索手段と、前記実行中スレッドリストにスレッドの識別情報が設定されている場合、前記クラスデータの削除を中止し、前記クラス管理オブジェクトの削除フラグをオンに設定する削除フラグ設定手段とを備えることを特徴とする請求項21に記載の情報処理装置。

【請求項23】 前記外部公開メソッドは、前記エージェントクラスオブジェクトのクラスデータを置換するクラスデータ置換手段を備えることを特徴とする請求項13～22のいずれか1項に記載の情報処理装置。

【請求項24】 前記クラスデータ置換手段は、置換対象のクラスデータを管理しているクラス管理オブジェクトの実行中スレッドリストを検索する第4実行中スレッドリスト検索手段と、前記実行中スレッドリストにスレッドの識別情報が設定されている場合、前記クラスデータの置換を中止し、前記クラス管理オブジェクトの置換クラスデータ記憶領域に新たなクラスデータを設定する置換データ設定手段とを備えることを特徴とする請求項23に記載の情報処理装置。

【請求項25】 前記クラス管理オブジェクトの被参照オブジェクトリストが空でない場合、エラーを通知するエラー通知手段と、前記クラス管理オブジェクトの参照オブジェクトリストが空でない場合、削除対象のクラスデータの参照先のクラスデータを管理しているクラス管理オブジェクトの被参照オブジェクトリストから、前記クラス管理オブジェクトの識別情報を削除する識別情報削除手段とを備える

ことを特徴とする請求項21～24のいずれか1項に記載の情報処理装置。

【請求項26】 前記外部公開メソッドは、他のエージェントクラスオブジェクトから送られたクラスデータを実行するクラスデータ遠隔実行手段を備えることを特徴とする請求項13～25のいずれか1項に記載の情報処理装置。

【請求項27】 前記クラスデータ遠隔実行手段は、クラス管理オブジェクトの局所クラスデータテーブルに前記クラスデータを設定するクラスデータ設定手段を備えることを特徴とする請求項26に記載の情報処理装置。

【請求項28】 前記クラスデータ遠隔実行手段は、クラスデータ及び前記クラスデータから生成されたオブジェクトのシリアル化データを受信するデータ受信手段と、

前記クラスデータ及び前記シリアル化データに基づいて、前記クラスデータから生成されたオブジェクトを復元するオブジェクト復元手段とを備えることを特徴とする請求項26または27に記載の情報処理装置。

【請求項29】 前記クラスデータ遠隔実行手段は、実行対象のクラスデータが参照しているクラスデータを前記共有クラスデータテーブルから呼び出すクラスデータ呼び出し手段と、

前記実行対象のクラスデータとともに前記実行対象のクラスデータが参照しているクラスデータを、他のエージェントクラスオブジェクトに送信する第1クラスデータ送信手段とを備えることを特徴とする請求項26～28のいずれか1項に記載の情報処理装置。

【請求項30】 前記クラスデータ遠隔実行手段は、実行対象のクラスデータが、送信先のエージェントクラスオブジェクトの共有クラスデータテーブルに存在しているかどうかを検索する重複クラス検索手段と、

前記検索結果に基づいて、送信先のエージェントクラスオブジェクトに存在しているクラスデータを除外してから、前記送信先のエージェントクラスオブジェクトに実行対象のクラスデータを送信する第2クラスデータ送信手段とを備えることを特徴とする請求項26～29のいずれか1項に記載の情報処理装置。

【請求項31】 プログラムを記憶するプログラム記憶手段と、前記プログラムの使用状況を前記プログラムの機能ごとに監視するプログラム監視手段と、

前記プログラムの機能の使用状況に基づいて、前記プログラムの機能に対する処理を行うプログラム処理手段とを備えることを特徴とする情報処理装置。

【請求項32】 前記プログラム監視手段は、前記プログラムの機能を特定するための名前を管理する名前管理手段を備えることを特徴とする請求項31に記載の情報処理装置。

【請求項33】 前記プログラム処理手段は、使用中の

機能に対する処理要求が行われた場合、前記機能の使用が終了するまで、前記処理要求を一時的に保留することを特徴とする請求項31または32に記載の情報処理装置。

【請求項34】 前記処理要求は、前記プログラムの一部の機能に対する削除要求または置換要求であることを特徴とする請求項33に記載の情報処理装置。

【請求項35】 クラスデータから構成される機能の使用状況を監視するステップと、

10 前記クラスデータから構成される機能の使用状況に基づいて、前記クラスデータに対する処理を実行するかどうかを決定するステップとを備えることを特徴とする情報処理方法。

【請求項36】 複数の機能からなるプログラムの実行状況を、それぞれの機能ごとに監視するステップと、前記プログラムの機能の実行状況に基づいて、前記プログラムの機能に対する処理を行うステップとを備えることを特徴とする情報処理方法。

20 【請求項37】 前記プログラムの実行中に実行されない一部の機能がある場合、前記実行されない一部の機能の削除または置換を、前記プログラムの実行中に許可することを特徴とする請求項36に記載の情報処理方法。

【請求項38】 前記プログラムの実行中に実行が終了した一部の機能がある場合、前記実行が終了した一部の機能の削除または置換を、前記プログラムの実行中に許可することを特徴とする請求項36に記載の情報処理方法。

30 【請求項39】 前記プログラムの実行中の機能または実行予定のある機能に対して削除要求または置換要求があった場合、前記削除要求または前記置換要求があった機能の実行後に、前記機能の削除または置換を行うことを特徴とする請求項36に記載の情報処理方法。

【請求項40】 少なくとも第1の機能と第2の機能とを含むプログラムの実行状況を、それぞれの機能ごとに監視するステップと、

前記プログラムの実行中に前記第1の機能の削除要求を行うステップと、

40 前記プログラムが前記第2の機能を実行している時に、前記第1の機能を前記プログラムから削除するステップとを備えることを特徴とする情報処理方法。

【請求項41】 少なくとも第1の機能と第2の機能とを含むプログラムの実行状況を、それぞれの機能ごとに監視するステップと、

前記プログラムの実行中に前記第1の機能の置換要求を行うステップと、

前記プログラムが前記第2の機能を実行している時に、前記プログラムの前記第1の機能を置換するステップとを備えることを特徴とする情報処理方法。

50 【請求項42】 オブジェクトに追加されたクラスデータを記憶する機能と、

前記クラスデータの使用状況を監視する機能と、  
前記クラスデータの使用状況に基づいて、前記クラスデータの処理を行う機能とをコンピュータに実行させるプログラムを格納した前記コンピュータが読み取り可能な記憶媒体。

【請求項43】 クラスデータの使用状況を監視するエージェントクラスオブジェクトを生成するためのエージェントクラスを格納する機能と、  
前記クラスデータに対する処理が行われる場合、前記エージェントクラスに基づいて前記エージェントクラスオブジェクトを生成する機能とをコンピュータに実行させるプログラムを格納した前記コンピュータが読み取り可能な記憶媒体。

【請求項44】 プログラムを記憶する機能と、  
前記プログラムの使用状況を監視する機能と、  
前記プログラムの中に使用されていない一部のプログラムコードが存在する場合、前記使用されていない一部のプログラムコードについて処理を行う機能とをコンピュータに実行させるプログラムを格納した前記コンピュータが読み取り可能な記憶媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、情報処理装置に関し、特に、分散オブジェクト指向プログラミングを行う場合などに適用して好適なものである。

【0002】

【従来の技術】 従来、分散オブジェクト指向技術であるCORBA (Common Object Request Broker Architecture)、DCOM (Distributed Component Object Model)、JavaRMI (Java Remote Method Invocation) などを利用することにより、ローカルに存在するオブジェクトに対してメソッドを発行するのとはほとんど同じ手続で、遠隔のコンピュータに存在するオブジェクトに対してメソッドを発行することが可能なことから、分散システムを比較的容易に開発することが可能だった。

【0003】

【発明が解決しようとする課題】 しかしながら、従来の分散オブジェクト指向技術では、動作中のサーバオブジェクトの一部のプログラムコードを置き換える機能や、一部のプログラムコードを遠隔実行する機能がないため、分散オブジェクトの運用を柔軟に行うことができなかった。

【0004】 例えば、データベースサーバオブジェクトにおいて、新たな問い合わせコマンドに対応するプログラムを実行する場合には、そのサーバオブジェクトを終了させ、新たに追加するプログラムをサーバオブジェクトが使用できるように設定を変更した後、サーバオブジ

ェクトを再起動しなければならなかった。

【0005】 このため、分散システムにおけるサーバオブジェクトのプログラムコードの追加・更新には一般に長い時間を要することから、更新されるサービスに関連するサーバオブジェクトの数が多くなるにつれて、そのサービスを停止しなければならない時間が長くなるという問題があった。

【0006】 また、同一のオブジェクトに対して複数のメソッド呼出しを行い、それらの結果に依存して次のメソッド呼出しのパラメータを決定する場合や、メソッド呼出しのパラメータや戻り値のデータ量が大きい場合には、メソッド呼出しによるネットワーク遅延の影響が大きく、分散システムの利点が生かされないという問題があった。

【0007】 また、サーバオブジェクトをある程度汎用的な目的に使えるようにしたり、サーバの動作を呼出し側のプログラムでカスタマイズできるようにするためには、サーバオブジェクトのメソッドインタフェースを増やして細分化しなければならなかった。

【0008】 このため、従来の分散システムでは、インタフェースが複雑になって分かりにくいものになるだけでなく、1つの目的を達成するために呼び出さなければならないメソッドの数が増加し、サービスの応答速度がネットワーク遅延により劣化するという問題があった。

【0009】 さらに、従来の分散システムでは、プログラムコードの追加・更新を行う場合、インタフェースのセキュリティを確保するための保守・運用に関わる管理者の負担が大きいという問題があった。

【0010】 そこで、本発明の第1の目的は、動作中のプログラムを停止させることなく、そのプログラムコードの一部を削除したり、置き換えたりすることを可能とすることである。

【0011】 また、本発明の第2の目的は、プログラムにクラス管理の知識がなくても、プログラムを容易にカスタマイズできるようにすることである。また、本発明の第3の目的は、分散システムにおけるプログラムコードの追加・更新のインターフェイスのセキュリティを向上させることである。

【0012】

【課題を解決するための手段】 上述した課題を解決するために、本発明によれば、オブジェクトに追加されたクラスデータを記憶するクラスデータ記憶手段と、クラスデータの使用状況を監視するクラスデータ監視手段と、クラスデータの使用状況に基づいて、クラスデータの処理を行うクラスデータ処理手段とを備えている。

【0013】 このことにより、プログラムをクラスデータで構成することが可能となるとともに、プログラムの使用状況を追加されたクラスデータごとに監視することが可能となり、プログラムが動作している場合にどのクラスデータが使用されているかを判別することが可能と

なることから、そのプログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となる。

【0014】また、本発明の一態様によれば、処理要求があったクラスデータが使用中の場合、そのクラスデータの使用が終了するまで、そのクラスデータの処理を延期するようにしている。

【0015】このことにより、動作中のプログラムに対して削除や置換などの処理要求が行われた場合においても、そのプログラムの動作を停止させることなく、そのプログラムの一部の機能に対する削除や置換などの処理を行わせることが可能となり、プログラムのカスタマイズ時におけるプログラムの負担を軽減することが可能となる。

【0016】また、本発明の一態様によれば、クラスデータについての処理要求を受け付けるかどうかを検査する検査手段を備えている。このことにより、プログラムコードの追加要求や更新要求を却下することが可能となり、プログラムコードの追加・更新のインターフェイスのセキュリティを向上させることが可能となる。

【0017】また、本発明の一態様によれば、クラスデータの追加要求単位でクラスデータをクラスデータ記憶手段に追加するようにしている。このことにより、クラスデータの追加要求を識別することが可能となり、追加要求時のクラスデータからなる機能単位でプログラムに対する処理を行うことが可能となることから、プログラマにクラス管理の知識がなくても、プログラムを容易にカスタマイズすることが可能となる。

【0018】また、本発明の一態様によれば、クラスデータの使用状況を監視するエージェントクラスオブジェクトを生成するためのエージェントクラスを設け、クラスデータに対する処理が行われる場合、エージェントクラスに基づいてエージェントクラスオブジェクトを生成するようにしている。

【0019】このことにより、監視対象となるプログラム自体にそのプログラムを監視する機能を容易に付加することが可能となり、プログラマにクラス管理の知識がなくても、プログラムの使用状況の監視を容易に行うことが可能となるとともに、そのプログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となる。

【0020】また、本発明の一態様によれば、エージェントクラスは、エージェントクラスオブジェクトに追加されたクラスデータの管理を行うクラス管理オブジェクトを、クラスデータの追加要求が行われることに生成するようにしている。

【0021】このことにより、プログラムに新たに追加された機能ごとにプログラムの監視を行うことが可能となり、プログラムが動作している場合にプログラムのどの機能が使用されているのかを判別することが可能となることから、プログラムの一部の機能を特定するだけ

で、そのプログラムの機能の削除や置換をそのプログラムの動作中に行うことが可能となる。

【0022】また、本発明の一態様によれば、クラスデータの追加要求に対して生成されたクラス管理オブジェクトとクラス管理オブジェクトの識別情報との対応関係を記憶するようにしている。

【0023】このことにより、クラス管理オブジェクトに与えられた名前などの識別情報を指定するだけで、プログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となり、プログラムをカスタマイズする際のプログラムの負担を軽減することが可能となる。

【0024】また、本発明の一態様によれば、クラス管理オブジェクトは、そのクラス管理オブジェクトに固有のクラスデータを格納する局所クラスデータテーブルを備えている。

【0025】このことにより、他のオブジェクトから送られてきたクラスデータとクラス管理オブジェクトが既に管理しているクラスデータとの間に衝突が発生した場合においても、他のオブジェクトから送られてきたクラスデータをクラス管理オブジェクトが既に管理しているクラスデータと独立に保有することが可能となり、エージェントクラスオブジェクトは、他のオブジェクトから送られてきたクラスデータを遠隔実行することが可能となる。

【0026】また、本発明の一態様によれば、クラス管理オブジェクトは、クラスデータの削除要求を記憶する削除フラグを備えている。このことにより、クラスデータに対する削除要求を記憶しておくことが可能となり、削除要求されたクラスデータが使用中の場合においても、そのクラスデータに対する削除要求を受け入れることが可能となることから、クラスデータに対する削除要求が拒否されたために、そのクラスデータに対する削除要求を改めて行う必要がなくなり、プログラマがクラスデータを削除する際の負担を軽減することが可能となる。

【0027】また、本発明の一態様によれば、クラス管理オブジェクトは、クラスデータの置換要求時の新たなクラスデータを記憶する置換クラスデータ記憶領域を備えている。

【0028】このことにより、置換要求時の新たなクラスデータを記憶しておくことが可能となり、置換要求されたクラスデータが使用中の場合においても、そのクラスデータに対する置換要求を受け入れることが可能となることから、クラスデータに対する置換要求が拒否されたために、そのクラスデータに対する置換要求を改めて行う必要がなくなり、プログラマがクラスデータを置換する際の負担を軽減することが可能となる。

【0029】また、本発明の一態様によれば、クラス管理オブジェクトは、クラスデータを実行しているスレッ



ドの識別情報を記憶する実行中スレッドリストを備えている。

【0030】このことにより、実行中スレッドリストを参照するだけで、実行されているクラスデータが存在するかどうかを判別することができ、クラスデータの使用状況を容易に把握することが可能となる。

【0031】また、本発明の一態様によれば、クラス管理オブジェクトは、追加されたクラスデータに対応するクラス名を記憶するクラス名リストを備えている。このことにより、クラス管理オブジェクトは、自己が管理しているクラスデータを容易に把握することが可能となり、追加されたクラスデータを単位としてクラスデータに対する処理を行うことが可能となる。

【0032】また、本発明の一態様によれば、クラス管理オブジェクトは、クラスデータの参照先のクラスデータを管理しているクラス管理オブジェクトの識別情報を記憶する参照オブジェクトリストと、クラスデータの参照元のクラスデータを管理しているクラス管理オブジェクトの識別情報を記憶する被参照オブジェクトリストとを備える。

【0033】このことにより、クラスデータの参照関係を考慮しながらクラスデータに対する処理を行うことが可能となり、プログラムがクラスデータの参照関係を認識する必要がなくなることから、プログラムのカスタマイズを容易に行うことが可能となる。

【0034】また、本発明の一態様によれば、エージェントクラスは、他のエージェントクラスオブジェクトからのアクセス権をチェックするアクセス制御クラスを備えている。

【0035】このことにより、エージェントクラスオブジェクトは、他のエージェントクラスオブジェクトからのアクセスがあった場合、アクセス制御クラスに基づいてアクセス制御オブジェクトを生成することが可能となることから、エージェントクラスオブジェクトからのアクセスがある度に、アクセス権のチェックを行うことが可能となるとともに、エージェントクラスオブジェクトごとにアクセス条件を容易に変更することが可能となり、プログラムコードの追加・更新のインターフェイスのセキュリティを向上させることが可能となる。

【0036】また、本発明の一態様によれば、エージェントクラスオブジェクトにクラスデータを追加する外部公開メソッドを備えている。このことにより、プログラムの実行中においても、そのプログラムに対してクラスデータを新たに追加することが可能となる。

【0037】また、本発明の一態様によれば、エージェントクラスオブジェクトに追加されたクラスデータに対応するクラス管理オブジェクトを生成するクラス管理オブジェクト生成手段と、クラス管理オブジェクト内のクラス名リストにクラスデータのクラス名を登録するクラス名登録手段と、クラスデータを前記共有クラスデータ

テーブルに書き込むクラスデータテーブル書き込み手段と、クラス管理オブジェクトの識別情報を前記クラス管理オブジェクトテーブルに登録するクラス管理オブジェクトテーブル登録手段とを備えている。

【0038】このことにより、エージェントクラスオブジェクトは、追加されたクラスデータごとにその使用状況を監視することが可能となるとともに、追加されたクラスデータを容易に特定してそのクラスデータに対する処理を実行することが可能となる。

10 【0039】また、本発明の一態様によれば、エージェントクラスオブジェクトに追加されたクラスデータのクラス名が、共有クラスデータテーブルに登録されているクラスデータのクラス名と重複する場合、エージェントクラスオブジェクトに追加されたクラスデータを共有クラスデータテーブルから削除するようにしている。

20 【0040】このことにより、共有クラスデータテーブル内で同一のクラスの衝突が発生することを防止することができる。また、本発明の一態様によれば、エージェントクラスオブジェクトに保有されているクラスデータを実行する外部公開メソッドを備えている。

【0041】このことにより、エージェントクラスオブジェクトはプログラムを実行させることが可能となる。また、本発明の一態様によれば、実行中のクラスデータが他のクラスデータを参照している場合、実行中のクラスデータを管理しているクラス管理オブジェクトに実行情報を設定するとともに、参照先のクラスデータを管理しているクラス管理オブジェクトにも実行情報を設定するようにしている。

30 【0042】このことにより、参照元のクラスデータの実行中に参照先のクラスデータの削除要求や置換要求がなされても、参照元のクラスデータの実行が終了するまで参照先のクラスデータの削除や置換を一時的に保留しておくことが可能となり、クラスデータの実行が途中で不可能になることを防止することが可能となる。

40 【0043】また、本発明の一態様によれば、実行中スレッドリストが空で、かつ、削除フラグがオンのクラス管理オブジェクトが存在している場合、そのクラス管理オブジェクトのクラス名リストのクラス名に対応するクラスデータを共有クラスデータテーブルから削除するようにしている。

【0044】このことにより、削除要求があったプログラムの一部の機能について、そのプログラムの一部の機能の実行が終了した後に、そのプログラムの一部の機能を削除することが可能となることから、プログラムの一部の機能を削除する際に、実行中のプログラムを途中で停止させたり、そのプログラムを再起動させたりする時間を省くことが可能となる。

50 【0045】また、本発明の一態様によれば、実行中スレッドリストが空で、かつ、置換クラスデータ記憶領域にクラスデータが記憶されている場合、共有クラスデー



テーブルに記憶されている置換対象のクラスデータを、置換クラスデータ記憶領域に記憶されているクラスデータで置換するようにしている。

【0046】このことにより、置換要求があったプログラムの一部の機能について、そのプログラムの一部の機能の実行が終了した後に、そのプログラムの一部の機能を他の機能で置換することが可能となることから、プログラムの一部の機能を置換する際に、実行中のプログラムを途中で停止させたり、そのプログラムを再起動させたりする手間を省くことが可能となる。

【0047】また、本発明の一態様によれば、エージェントクラスオブジェクトからクラスデータを削除する外部公開メソッドを備えている。このことにより、エージェントクラスオブジェクトはプログラムの一部を削除することが可能となる。

【0048】また、本発明の一態様によれば、クラス管理オブジェクトの実行中スレッドリストにスレッドの識別情報が設定されている場合、そのスレッドで実行されているクラスデータの削除を中止し、そのクラス管理オブジェクトの削除フラグをオンに設定するようにしている。

【0049】このことにより、実行中のクラスデータに対して削除要求が行われた場合においても、そのクラスデータの実行を停止させることなく、そのクラスデータに対する削除要求を受け入れることが可能となることから、そのクラスデータの実行に影響を与えることなく、そのクラスデータを削除することが可能となる。

【0050】また、本発明の一態様によれば、エージェントクラスオブジェクトのクラスデータを置換する外部公開メソッドを備えている。このことにより、エージェントクラスオブジェクトはプログラムの一部を置換することが可能となる。

【0051】また、本発明の一態様によれば、クラス管理オブジェクトの実行中スレッドリストにスレッドの識別情報が設定されている場合、そのスレッドで実行されているクラスデータの置換を中止し、そのクラス管理オブジェクトの置換クラスデータ記憶領域に新たなクラスデータを設定するようにしている。

【0052】このことにより、実行中のクラスデータに対して置換要求が行われた場合においても、そのクラスデータの実行を停止させることなく、そのクラスデータに対する置換要求を受け入れることが可能となり、そのクラスデータの実行に影響を与えることなく、そのクラスデータを置換することが可能となる。

【0053】また、本発明の一態様によれば、削除要求や置換要求に対し、クラス管理オブジェクトの被参照オブジェクトリストが空でない場合、エラーを通知し、クラス管理オブジェクトの参照オブジェクトリストが空でない場合、参照先のクラス管理オブジェクトの被参照オブジェクトリストから、参照元のクラス管理オブジェク

トの識別情報を削除するようにしている。

【0054】このことにより、他のクラスデータが参照しているクラスデータが消失することを防止することが可能となるとともに、不要となった情報を削除することが可能となり、プログラムをカスタマイズする際の利便性を向上させることが可能となる。

【0055】また、本発明の一態様によれば、他のエージェントクラスオブジェクトから送られたクラスデータを実行する外部公開メソッドを備えている。このことにより、エージェントクラスオブジェクトはプログラムの遠隔実行を行うことが可能となる。

【0056】また、本発明の一態様によれば、他のエージェントクラスオブジェクトから送られたクラスデータをクラス管理オブジェクトの局所クラスデータテーブルに格納するようにしている。

【0057】このことにより、クラス管理オブジェクトに対応して既に格納されているクラスデータとの衝突を回避することが可能となる、エージェントクラスオブジェクトによる遠隔実行を円滑に行うことが可能となる。

【0058】また、本発明の一態様によれば、クラスデータ及びクラスデータから生成されたオブジェクトのシリアルイズデータに基づいて、そのクラスデータから生成されたオブジェクトを復元するようにしている。

【0059】このことにより、クラスデータから生成されたオブジェクトを他のエージェントクラスオブジェクトに送信して、そのオブジェクトの遠隔実行を行うことが可能となる。

【0060】また、本発明の一態様によれば、実行対象のクラスデータとともにその実行対象のクラスデータが参照しているクラスデータを、他のエージェントクラスオブジェクトに送信するようにしている。

【0061】このことにより、参照関係を有しているクラスデータについても、遠隔実行を行うことが可能となる。また、本発明の一態様によれば、送信先のエージェントクラスオブジェクトに存在しているクラスデータを除外して、実行対象のクラスデータをその送信先のエージェントクラスオブジェクトに送信するようにしている。

【0062】このことにより、遠隔実行を行う際の通信量を減らすことが可能となり、遠隔実行の速度を上げることが可能となる。また、本発明の一態様によれば、複数の機能からなるプログラムの実行状況を、それぞれの機能ごとに監視し、そのプログラムの機能の実行状況に基づいて、そのプログラムの機能に対する処理を行うようにしている。

【0063】このことにより、プログラムの実行中に一部の機能が実行されない場合や、プログラムの実行中に一部の機能の実行が終了した場合、その一部の機能の削除または置換をそのプログラムの実行中に行うことが可能となり、プログラムのカスタマイズを迅速に行うこと

10

20

30

40

50

が可能となる。

【0064】

【発明の実施の形態】以下、本発明の実施例について図面を参照しながら説明する。図1は、本発明の一実施例に係わる情報処理装置の機能的な構成を示すブロック図である。

【0065】図1において、クラスデータ記憶手段1は、オブジェクトに追加されたクラスデータを記憶する。クラスデータ監視手段2は、クラスデータ記憶手段1に記憶されているクラスデータの使用状況を追加されたクラスデータごとに監視する。クラスデータ処理手段3は、クラスデータ監視手段2で監視されているクラスデータの使用状況に基づいて、クラスデータの処理を追加されたクラスデータごとに行う。

【0066】このため、クラスデータ監視手段2は、プログラムに新たに追加された機能ごとにプログラムの監視を行うことが可能となり、プログラムが動作している場合にプログラムのどの機能が使用されているのかを判別することが可能となる。この結果、クラスデータ処理手段3は、プログラムの一部の機能を特定するだけで、そのプログラムの機能に対する処理をそのプログラムの動作中に行うことが可能となる。

【0067】図2は、図1のクラスデータ処理手段3の構成例を示すブロック図である。図2において、受付手段11は、クラスデータ記憶手段1に記憶されているクラスデータについての削除要求や置換要求などを受け付ける。判定手段12は、クラスデータ監視手段2での監視結果に基づいて、削除要求や置換要求などがあったクラスデータが使用中かどうかを判定する。延期手段13は、クラスデータが使用中の場合、そのクラスデータの使用が終了するまで、そのクラスデータの削除や置換などを延期する。

【0068】ここで、受付手段11は、実行中のクラスデータに対して削除要求や置換要求が行われた場合、そのクラスデータの実行を停止させることなく、そのクラスデータに対する削除要求や置換要求を受け付ける。そして、遅延手段13は、判定手段12による判定結果を参照することにより、削除対象または置換対象となっているクラスデータの使用が終了したかどうかを識別し、削除対象または置換対象となっているクラスデータの使用が終了してから、受付手段11が受け付けた削除や置換などの処理を実行する。

【0069】このことにより、削除要求や置換要求があったプログラムの一部の機能について、そのプログラムの一部の機能の実行が終了した後に、そのプログラムの一部の機能を削除したり、他の機能で置換したりすることが可能となり、プログラムの一部の機能を削除したり置換したりする際に、実行中のプログラムを途中で停止させたり、そのプログラムを再起動させたりする手間を省くことが可能となる。

【0070】図3は、本発明の一実施例に係わるコンピュータの機能的な構成を示すブロック図である。図3において、開発言語のライブラリの基本クラスとしてエージェントクラスが設けられ、このエージェントクラスはエージェントクラスライブラリ31に格納されている。プログラマは、システム開発を行う場合、この基本クラスとしてのエージェントクラスに基づいて、サブクラスを生成することにより、アプリケーションを記述する。

【0071】エージェントクラスには、クラスデータの追加要求に対して生成されたクラス管理オブジェクト24a～24mとクラス管理オブジェクト24a～24mの名前との対応関係を記憶したクラス管理オブジェクトテーブル23、エージェントクラスオブジェクト22a～22nに追加されたクラスデータの管理を行うクラス管理オブジェクト24a～24mと、クラス管理オブジェクト24a～24mで管理されるクラスデータを記憶する共有クラスデータテーブル25及びクラスデータに対する処理を実行する外部公開メソッド26が設けられている。

【0072】エージェントクラスオブジェクト22a～22nは、エージェントクラスライブラリ31に格納されているエージェントクラスのインスタンスとして、コンピュータ21上に生成される。エージェントクラスオブジェクト22aのクラス管理オブジェクトテーブル23には、例えば、クラス管理オブジェクト24a～24mの名前“function-1”～“function-m”がそれぞれ登録されるとともに、それぞれの名前“function-1”～“function-m”に対応するクラス管理オブジェクト24a～24mへのポインタが格納されている。

【0073】エージェントクラスオブジェクト22aの外部公開メソッド26として、例えば、remove()メソッド26a、add()メソッド26b、get()メソッド26c、set()メソッド26d、eval()メソッド26e、call()メソッド26fが設けられている。

【0074】remove()メソッド26aは、削除対象となるクラスデータを管理しているクラス管理オブジェクト24a～24mの名前を引数として、共有クラスデータテーブル25からクラスデータを削除する。

【0075】add()メソッド26bは、追加対象となるクラスデータを管理するクラス管理オブジェクト24a～24mの名前及び追加対象となるクラスデータを引数として、追加対象となるクラスデータを管理するクラス管理オブジェクト24a～24mを新たに生成するとともに、そのクラス管理オブジェクト24a～24mの名前をクラス管理オブジェクトテーブル23に登録し、追加対象となるクラスデータを共有クラスデータテーブル25に格納する。

【0076】get()メソッド26cは、取得対象と

なるクラスデータを管理しているクラス管理オブジェクト24a~24mの名前を引数として、共有クラスデータテーブル25からクラスデータをコピーする。

【0077】set()メソッド26dは、置換対象となるクラスデータを管理しているクラス管理オブジェクト24a~24mの名前及び置換する新たなクラスデータを引数として、共有クラスデータテーブル25の置換対象となるクラスデータを新たなクラスデータで置換する。

【0078】eval()メソッド26eは、実行対象となるクラスデータ及びそのクラスデータのオブジェクトのシリアル化データを引数として、そのクラスデータまたはそのオブジェクトについての遠隔実行を行う。

【0079】call()メソッド26fは、実行対象となるクラスデータを管理しているクラス管理オブジェクト24a~24mの名前を引数として、共有クラスデータテーブル25に格納されているクラスデータからオブジェクトを生成して実行する。

【0080】コンピュータ21には、また、通信デモン28、ネットワークソケットAPI(アプリケーションプログラミングインターフェイス)29、オペレーティング・システム30及びシステムクラスライブラリ32が設けられている。

【0081】外部公開メソッド26が呼ばれた際には、通信スレッド27a~27kが割り当てられ、1つのプログラム中で複数の実行の流れ(例えば、メソッドや関数)が、データ領域を共有しながら並列して行われることが可能となっている。なお、このマルチスレッド機能は、Java言語などでは標準的に備わっているものである。

【0082】コンピュータ21内のエージェントクラスオブジェクト22a~22n間での情報の通信や、他のコンピュータとの間で情報の通信は、ネットワークソケットAPIを介して行われる。

【0083】エージェントクラスオブジェクト22a~22nのクラス置換機能や遠隔実行機能が実際のシステムで利用される状況としては、例えば、ソフトウェアの遠隔保守が挙げることができる。そして、これらのクラス置換機能や遠隔実行機能を利用することにより、サーバを停止させることなく、ソフトウェアの入れ替えを遠隔に行うことが可能になるとともに、複数のサーバをバッチ処理的に行うことができるようになる。

【0084】このため、サーバソフトの一部だけを対象として、プログラムを変更する場合でも、サーバを停止し、プログラムを変更した後に、サーバを再起動するような手間をなくすることが可能となり、特に、分散システムの場合には、管理者の負担を大幅に軽減することが可能となる。また、一般的なアプリケーションについても、エージェントクラスオブジェクト22a~22n内部で動作するクラスとして作成することにより、アプリ

ケーションのカスタマイズをより柔軟に行うことが可能となる。

【0085】このように、図3の実施例では、複数のエージェントクラスオブジェクト22a~22nが連携し、動作中のエージェントクラスオブジェクト22a~22nが、そのエージェントクラスオブジェクト22a~22nを構成するプログラムコードであるクラスデータの追加要求または削除要求を受け付ける場合に、それぞれのエージェントクラスオブジェクト22a~22n内に共有クラスデータテーブル25を設け、追加要求の発生ごとにクラス管理オブジェクト24a~24mを生成する。

【0086】そして、そのクラス管理オブジェクト24a~24mが、共有クラスデータテーブル25の検索や追加されたクラス群の実行及び削除を管理することにより、追加されたクラス群を共有することが可能となるとともに、部分的なクラス群を動的に削除・更新することが可能となる。

【0087】また、ランタイム環境におけるプログラムの置換機能やプログラムの一部の遠隔実行を利用することにより、分散型ネットワーク管理システム、分散型ネットワークサービス、分散型グループウェアなどでの柔軟な相互連携や動的な負荷分散が可能となる。

【0088】さらに、動作中のサーバオブジェクトのプログラムコードの追加・更新をサーバオブジェクトの動作中に行うことが可能となり、更新されるサービスに関連するサーバオブジェクトを完全に停止させる必要がなくなることから、サービスの停止期間を飛躍的に短くすることが可能となる。

【0089】なお、オブジェクトを構成するクラスデータを監視する機能をそのオブジェクトに設けたものをエージェントと呼ぶ。図4は、図3のクラス管理オブジェクトの構成例を示す図である。

【0090】図4において、クラス管理オブジェクト24a~24mには、局所クラスデータテーブル(local classes)41、削除フラグ(removeable flag)42、置換クラスデータ記憶領域(next classes)43、実行中スレッドIDリスト(running threads)44、クラス名リスト(class names)45、参照オブジェクトリスト(referring)46、被参照オブジェクトリスト(referred)47及びクラスデータの検索を行うクラス探索機能48が設けられている。

【0091】局所クラスデータテーブル41は、プログラムの遠隔実行時に受信したクラスデータを保存するもので、遠隔実行時に他のエージェントクラスオブジェクト22a~22nからクラスデータが送られてきた場合、そのクラスデータをクラス管理オブジェクト24a~24mにより既に管理されているクラスデータと独立

に保有することにより、送信されたクラスデータとクラス管理オブジェクト24a~24mが既に管理しているクラスデータとの間の衝突を防止することを可能とし、クラスデータの遠隔実行を行うことを可能とするものである。

【0092】削除フラグ42は、共有クラスデータテーブル25に追加されたクラスデータの削除要求が発生したかどうかを示すもので、クラスデータに対する削除要求を削除フラグ42に記憶しておくことにより削除要求されたクラスデータが使用中の場合においても、その

クラスデータに対する削除要求を受け入れることを可能とするものである。

【0093】置換クラスデータ記憶領域43は、共有クラスデータテーブル25に追加されたクラスデータの置換要求が発生した場合に、置換する新しいクラスデータを一時的に保存しておくもので、置換要求時の新たなクラスデータを置換クラスデータ記憶領域43に記憶しておくことにより、置換要求されたクラスデータが使用中の場合においても、そのクラスデータに対する置換要求を受け入れることを可能とするものである。

【0094】実行中スレッドリスト44は、共有クラスデータテーブル25に追加されたクラスデータを実行しているスレッドIDを記憶するもので、実行中スレッドリスト44にスレッドIDを記憶することにより、実行中スレッドリスト44を参照するだけで、実行されているクラスデータが存在するかどうかを判別することを可能とするものである。

【0095】クラス名リスト45は、共有クラスデータテーブル25に追加されたクラスデータの各クラス名を記憶するもので、クラス管理オブジェクト24a~24mが管理しているクラスデータを把握することを可能とし、クラス管理オブジェクト24a~24mが管理するクラスデータを単位として処理を行うことを可能とするものである。

【0096】参照オブジェクトリスト46は、共有クラスデータテーブル25に追加されたクラスデータが利用する他のクラスデータを、どのクラス管理オブジェクト24a~24mが管理しているかを記憶するもので、参照先のクラスデータを考慮しながらクラスデータに対する処理を行うことを可能とするものである。

【0097】被参照オブジェクトリスト47は、クラスデータを利用している他のクラスデータを、どのクラス管理オブジェクト24a~24mが管理しているかを記憶するもので、参照元のクラスデータを考慮しながらクラスデータに対する処理を行うことを可能とするものである。

【0098】図5は、エージェントクラスの構成例を示す図である。エージェントクラスは、クラスライブラリ51の基本クラス52として設けられ、この基本クラス52には、クラス管理オブジェクトテーブル53、クラ

ス管理オブジェクト5、共有クラスデータテーブル55及び外部公開メソッド56が設けられている。また、クラスライブラリ51には、基本クラス52から生成されたサブクラス57が登録され、エージェントクラスオブジェクト22a~22nは、サブクラス57からのインスタンス58a~58jとして生成することができる。

【0099】このように、分散システムの開発者は、エージェントクラスのサブクラス57を作成して、アプリケーションを記述することにより、クラス管理に関する特別な知識がなくとも、ランタイム環境でのクラスデータの置換機能を実現したり、外部のエージェントクラスオブジェクト22a~22nからクラスデータを送り付けることにより、遠隔実行機能を利用した柔軟なシステムを構築することができる。

【0100】図6は、クラスデータ追加時のクラス管理オブジェクトの生成方法を示す図である。図6において、例えば、“function-1”という名前前でクラスデータA、クラスデータB及びクラスデータCをエージェントクラスオブジェクト301に追加する場合、

クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル302に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト303が生成される。

【0101】クラス管理オブジェクト303には、クラス管理オブジェクト303が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名がそれぞれ登録されるとともに、クラス管理オブジェクト303の名前として、“function-1”がクラス管理オブジェクトテーブルに登録される。

【0102】クラスデータA、クラスデータB及びクラスデータCがエージェントクラスオブジェクト301に追加された後、“function-2”という名前前でクラスデータD及びクラスデータEをエージェントクラスオブジェクト301に追加する場合、クラスデータD及びクラスデータEが共有クラスデータテーブル302に格納されるとともに、クラスデータD及びクラスデータEを管理するクラス管理オブジェクト304が新たに生成される。

【0103】クラス管理オブジェクト304には、クラス管理オブジェクト304が管理するクラスデータD及びクラスデータEのクラス名がそれぞれ登録されるとともに、クラス管理オブジェクト304の名前として、“function-2”がクラス管理オブジェクトテーブルに登録される。

【0104】クラスデータD及びクラスデータEがエージェントクラスオブジェクト301に追加された後、“function-3”という名前前でクラスデータF、クラスデータG及びクラスデータHをエージェントクラスオブジェクト301に追加する場合、クラスデー

10

20

30

40

50

タF、クラスデータG及びクラスデータHが共有クラスデータテーブル302に格納されるとともに、クラスデータF、クラスデータG及びクラスデータHを管理するクラス管理オブジェクト305が新たに生成される。

【0105】クラス管理オブジェクト305には、クラス管理オブジェクト305が管理するクラスデータF、クラスデータG及びクラスデータHのクラス名がそれぞれ登録されるとともに、クラス管理オブジェクト305の名前として、“function-3”がクラス管理オブジェクトテーブルに登録される。

【0106】このように、クラスデータにより構成される機能が、エージェントクラスオブジェクト301に追加されるごとに、クラス管理オブジェクト303～305を生成することにより、クラスデータにより構成される機能ごとに、プログラムの監視を行うことが可能となる。このため、プログラムが動作している場合にプログラムのどの機能が使用されているのかを判別することが可能となり、そのプログラムの一部の機能の削除や置換をそのプログラムの動作中に行うことが可能となる。

【0107】また、クラス管理オブジェクト303～305の名前“function-1”～“function-3”をクラス管理オブジェクトテーブルに登録しておくことにより、クラス管理オブジェクト303～305の名前“function-1”～“function-3”を指定するだけで、プログラマがプログラムに追加した機能を特定することが可能となり、プログラムをカスタマイズする際の利便性を向上させることが可能となる。

【0108】次に、本発明の一実施例に係わるアプリケーションのカスタマイズ方法について説明する。図7は、テンプレートのコード化の例を示す図である。

【0109】図7において、例えば、Eメールの着信通知をスクリーニングするアプリケーションが、初期設定ファイルを使ったテンプレート（型枠）形式で提供されているものとする。このテンプレートは、特定の発信者からのメールの着信通知にスクリーニングをかけるためのもので、初期設定メニュー81のフィールド欄83のポップアップメニュー82から“From”を選択し、スクリーニングをかける発信者の名称をキーボードから入力することにより、発信者の名称を内容欄84に設定するようになっている。

【0110】一方、Eメール本文の中からキーワードを検索してスクリーニングを行う場合、図7のテンプレートにより提供される機能を用いただけでは、このキーワード検索を行うことは不可能である。このため、Eメール自体のテキストをキーワード検索するプログラムコードを外部から持ってくる。そして、図7のテンプレートにより提供されるEメールの着信通知の部分のプログラムコードを、Eメール自体のテキストをキーワード検索するプログラムコードと置換する。

【0111】ここで、このプログラムコードの置換を行う際には、クラス管理オブジェクト24a～24mが、Eメールの着信通知の部分のプログラムコードの使用状況を監視することにより、Eメールの着信通知の部分のプログラムコードが実行されていない時に、プログラムコードの置換を行わせることが可能となる。このため、図7のテンプレートをカスタマイズする際に、図7のテンプレート形式によるプログラムの動作を停止させることなく、図7のテンプレート形式によるプログラムの機能の一部を置換することが可能となる。

【0112】このように、テンプレートをコード化するとともに、プログラムコードを機能ごとに監視することにより、カスタマイズするプログラムを動作させながら、構築するアプリケーションに必要なプログラムコードを検索して置換することが可能となり、手元に備わっているテンプレートに制約されることなく、アプリケーションのカスタマイズを柔軟に行うことが可能となる。

【0113】図8は、図3のコンピュータのシステム構成例を示す図である。図8において、CPU61、RAM62、ROM63、通信インターフェイス64、プリンタ65、ディスプレイ66、キーボード67、マウス68及びドライバ69が互いに接続され、ドライバ69には、例えば、ハードディスク70、フロッピーディスク71、磁気テープ72、CD-ROMやDVD-ROMなどの光ディスク73、ICメモ리카ード74が接続されている。

【0114】エージェントクラスは、ROM63、ハードディスク70、フロッピーディスク71、磁気テープ72、CD-ROMやDVD-ROMなどの光ディスク73、ICメモ리카ード74などに格納され、エージェントクラスからエージェントクラスオブジェクト22a～22nが起動されると、CPU61の制御により、エージェントクラスオブジェクト22a～22nがRAM62上に生成される。ここで、エージェントクラスオブジェクト22a～22nに対する操作は、ディスプレイ66に表示されるテンプレート形式での画面操作やキーボード入力により行うことができる。

【0115】なお、プログラムを通信インターフェイス64を介して送受信することも可能であり、通信インターフェイス64と接続される通信ネットワークとして、例えば、LAN (Local Area Network)、WAN (Wide Area Network)、インターネット、アナログ電話網、デジタル電話網 (ISDN: Integral Service Digital Network)、PHS (パーソナルハンディシステム) などの無線通信網などを用いることができる。

【0116】図9は、図3の通信デーモン28の動作を示すフローチャートである。図9において、メッセージ受信待ちの状態（ステップS1）、通信デーモン28

が、他のコンピュータからのメッセージをネットワークソケットAPI 29を介して受信すると、転送先のエージェントクラスオブジェクト22a~22nの名前“AGENT-1”~“AGENT-n”、メソッド名及び引数をこのメッセージから取得し(ステップS3)。そして、他のコンピュータからの取得したデータを通信スレッド27a~27kに渡す。

【0117】図10は、図3の通信スレッド27a~27kの動作を示すフローチャートである。図10において、通信スレッド27a~27kが通信デーモン28から生成されると(ステップS11)、通信デーモン28により指定されたエージェントクラスオブジェクト22a~22nの名前“AGENT-1”~“AGENT-n”から、エージェントクラスオブジェクト22a~22nを特定する(ステップS12)。

【0118】次に、通信スレッド27a~27kは、特定したエージェントクラスオブジェクト22a~22nに対し、通信デーモン28から渡された引数を設定して外部公開メソッド26の呼び出しを行い(ステップS13)、エージェントクラスオブジェクト22a~22nから戻り値が返されるのを待つ(ステップS14)。

【0119】次に、エージェントクラスオブジェクト22a~22nから外部公開メソッド26の戻り値が返されたら、ネットワーク・ソケットAPI 29を制御して、外部公開メソッド26を発行した他のコンピュータのエージェントクラスオブジェクトに戻り値を含むメッセージを返送し(ステップS15)、通信スレッド27a~27kを消滅させる(ステップS16)。

【0120】次に、エージェントクラスオブジェクト22a~22nへのクラスデータの追加方法について説明する。図11は、クラスデータをエージェントクラスオブジェクトに追加した状態を示す図である。

【0121】図11において、例えば、“function-1”という名前でクラスデータA、クラスデータB及びクラスデータCからなる機能をエージェントクラスオブジェクト101に追加する場合、クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル104に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト103が生成され、クラス管理オブジェクト103の名前として、“function-1”がクラス管理オブジェクトテーブル102に登録される。そして、クラス管理オブジェクト103の名前“function-1”に対応して、クラス管理オブジェクト103を特定するポインタがクラス管理オブジェクトテーブル102に生成される。

【0122】また、クラス管理オブジェクト103のクラス名リスト(class names)には、クラス管理オブジェクト103が管理するクラスデータA、ク

ラスデータB及びクラスデータCのクラス名“A”、“B”、“C”がそれぞれ登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)、実行中スレッドIDリスト(running threads)、参照オブジェクトリスト(referring)及び被参照オブジェクトリスト(referred)が空(null)とされる。

10 【0123】図12は、参照関係のあるクラスデータをエージェントクラスオブジェクトに追加した状態を示す図である。図12において、例えば、“function-1”という名前でクラスデータA、クラスデータB及びクラスデータCからなる機能がエージェントクラスオブジェクト111に追加され、クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル112に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト113が生成されている。

20 【0124】このクラス管理オブジェクト113のクラス名リスト(class names)には、クラス管理オブジェクト113が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、“B”、“C”がそれぞれ登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)、実行中スレッドIDリスト(running threads)及び参照オブジェクトリスト(referring)が空(null)とされている。

30 【0125】また、クラス管理オブジェクト113の名前として、“function-1”がクラス管理オブジェクトテーブル112に登録され、クラス管理オブジェクト113の名前“function-1”に対応して、クラス管理オブジェクト113を特定するポインタがクラス管理オブジェクトテーブル112に生成されている。

40 【0126】この状態で、“function-3”という名前でクラスデータF及びクラスデータGからなる機能をエージェントクラスオブジェクト111に追加する場合、クラスデータF及びクラスデータGが共有クラスデータテーブル115に追加される。

50 【0127】また、クラスデータF及びクラスデータGを管理するクラス管理オブジェクト114が新たに生成され、クラス管理オブジェクト114の名前として、“function-3”がクラス管理オブジェクトテーブル112に登録され、クラス管理オブジェクト114の名前“function-3”に対応して、クラス管理オブジェクト114を特定するポインタがクラス管理オブジェクトテーブル112に新たに生成される。



【0128】さらに、クラス管理オブジェクト114のクラス名リスト(class names)には、クラス管理オブジェクト114が管理するクラスデータF及びクラスデータGのクラス名“F”、“G”がそれぞれ登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)、実行中スレッドIDリスト(running threads)及び被参照オブジェクトリスト(referred)が空(null)とされる。

【0129】ここで、クラスデータFが、例えば、

```
class F0 {
:
new A0 ;
:
new G0 ;
}
```

のように、共有クラスデータテーブル115に既に追加されているクラスデータAの存在を前提とする場合、クラス管理オブジェクト114の参照オブジェクトリスト(referring)には、参照先のクラスデータAを管理しているクラス管理オブジェクト113の参照情報が格納され、クラス管理オブジェクト113の被参照オブジェクトリスト(referred)には、参照元のクラスデータFを管理しているクラス管理オブジェクト114の被参照情報が格納される。

【0130】このため、名前“function-1”を引数として、remove()メソッド26aやset()メソッド26dが呼び出され、クラスデータA、クラスデータB及びクラスデータCからなる機能の削除要求や置換要求がエージェントクラスオブジェクト111に対して行われた場合でも、クラス管理オブジェクト113は、被参照オブジェクトリスト(referred)を調べることにより、クラスデータA、クラスデータB及びクラスデータCからなる機能の削除要求や置換要求を保留することが可能となり、クラスデータAを参照しているクラスデータFの機能が損なわれることを防止することが可能となる。

【0131】図13は、重複するクラスデータのエージェントクラスオブジェクト22a~22nへの登録を拒否する方法を示す図である。図13において、例えば、“function-1”という名前でもクラスデータA、クラスデータB及びクラスデータCからなる機能がエージェントクラスオブジェクト121に追加され、クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル122に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト123が生成されている。

【0132】このクラス管理オブジェクト123のクラス名リスト(class names)には、クラス管理オブジェクト123が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、“B”、“C”がそれぞれ登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)、実行中スレッドIDリスト(running threads)、参照オブジェクトリスト(referring)及び被参照オブジェクトリスト(referred)が空(null)とされている。

【0133】また、クラス管理オブジェクト123の名前として、“function-1”がクラス管理オブジェクトテーブル122に登録され、クラス管理オブジェクト123の名前“function-1”に対応して、クラス管理オブジェクト123を特定するポインタがクラス管理オブジェクトテーブル122に生成されている。

【0134】この状態で、名前“function-6”、クラスデータD、クラスデータE及びクラスデータFを引数として、エージェントクラスオブジェクト121のadd()メソッド125が呼ばれると、クラスデータDがエージェントクラスオブジェクト121の共有クラスデータテーブル124に格納されていないことを確認してから、共有クラスデータテーブル124にクラスデータDを新たに格納する。

【0135】次に、クラスデータEが共有クラスデータテーブル124に格納されていないことを確認してから、共有クラスデータテーブル124にクラスデータEを新たに格納する。

【0136】次に、共有クラスデータテーブル124にクラスデータAを格納する場合、クラスデータAは共有クラスデータテーブル124に既に格納されていることから、クラスデータD及びクラスデータEを共有クラスデータテーブル124から削除し、名前“function-6”のクラスデータD、クラスデータE及びクラスデータAについては、共有クラスデータテーブル124に格納しないこととする。

【0137】このように、エージェントクラスオブジェクト121が、複数のクラスデータを同時に追加したり、置換したりする要求を受ける時に、追加対象の一部のクラス名とエージェントクラスオブジェクト121内部に既に存在するクラス名との衝突がクラスデータの追加処理の途中で発生した場合、共有クラスデータテーブル124の追加対象のクラスデータを全て削除することにより、共有クラスデータテーブル124の元の状態を復元するようにしている。

【0138】図14は、クラスデータ追加処理を示すフローチャートである。ここで、エージェントクラスオブ



ジェクト22a~22nがクラスデータ追加処理を行う場合、add()メソッド26bが呼び出される。

【0139】図14において、まず、エージェントクラスオブジェクト22a~22nに対するアクセス権のチェックが行われ(ステップS21)、アクセス権がなければ処理が中断される。

【0140】次に、クラスデータの追加時に付ける名前“function-1”~“function-m”及び追加するクラスデータをadd()メソッド26bの引数から取得し(ステップS22)、クラスデータの追加時に付ける名前“function-1”~“function-m”をクラス管理オブジェクトテーブル23から検索する(ステップS23)。

【0141】次に、クラスデータの追加時に付ける名前“function-1”~“function-m”が、クラス管理オブジェクトテーブル23に既に存在するかどうかを調べ(ステップS24)、クラスデータの追加時に付ける名前“function-1”~“function-m”がクラス管理オブジェクトテーブル23に既に存在する場合、エラーを通知して処理を中断する(ステップS25)。

【0142】一方、クラスデータの追加時に付ける名前“function-1”~“function-m”がクラス管理オブジェクトテーブル23に存在しない場合、追加要求のあったクラスデータを共有クラスデータテーブル125に書き込む(ステップS26)。

【0143】次に、同一のクラス名が既に共有クラスデータテーブル25に存在しているかどうかを判断し(ステップS27)、クラスデータの書込みの途中で、同一のクラス名が既に共有クラスデータテーブル25に存在していた場合、共有クラスデータテーブル25に格納した追加要求時のクラスデータを削除し、共有クラスデータテーブル25を元の状態に戻すとともに(ステップS28)、エラーを通知して処理を終了する(ステップS\*

```
class CheckSet {
    void start (RemoteAgent
        requester,
        String name,
        Object val)
        throws AccessException {
        if (!requester.get ("name").
            equals ("Admin"))
            throw new AccessException ();
    }
}
```

そして、クラスデータの登録前に要求元のユーザ名、ホスト名、エージェントクラスオブジェクト22a~22nの名前“AGENT-1”~“AGENT-n”などからアクセス制御条件をチェックし、クラスデータに対する処理要求を受け付けるかどうかを決定する。すなわ

\*29)。

【0144】一方、クラスデータの書込みの途中で、同一のクラス名が既に共有クラスデータテーブル25に存在しなかったために、追加要求のあった全てのクラスデータの追加に成功した場合、クラス管理オブジェクト24a~24mを新たに生成する。そして、add()メソッド26bの引数に設定されていたクラスデータの追加時に付ける名前“function-1”~“function-m”をクラス管理オブジェクトテーブル23に登録し、新たに生成されたクラス管理オブジェクト24a~24mをクラス管理オブジェクトテーブル23に登録された名前“function-1”~“function-m”で識別できるようにする(ステップS30)。

【0145】次に、共有クラスデータテーブル25に追加した全てのクラスデータのクラス名を、新たに生成されたクラス管理オブジェクト24a~24mのクラス名リスト45に設定する。(ステップS31)。

【0146】次に、共有クラスデータテーブル25に追加したクラスデータに関連するクラスを調べ、新たに生成されたクラス管理オブジェクト24a~24mの参照オブジェクトリスト46に参照先情報を設定するとともに(ステップS32)、参照先のクラス管理オブジェクト24a~24mの被参照オブジェクトリスト47に参照元情報を設定して(ステップS33)、処理を終了する。

【0147】次に、本発明の一実施例に係わるアクセス権の制御方法について説明する。クラス置換時のアクセス権チェック用の処理を、例えば、“checkSet”という名前で予約し、以下のようなクラスを、“checkSet”という名前でエージェントクラスオブジェクト22a~22nに追加するものとする。

【0148】

ち、動作中のエージェントクラスオブジェクト22a~22nが、クラスデータの削除要求または置換要求を受け付ける場合に、名前“checkSet”のクラスからオブジェクトを生成して実行する。

【0149】この場合、クラスデータ置換要求の要求元

のエージェントの名前が“Admin”に一致しなければ、置換処理は行われずに、例外が発生する。このアクセス制御条件を記述したクラスは、エージェントクラスオブジェクト22a~22n内の他のクラスと扱いが同じであるため、運用中のアクセス制御条件の変更が可能となる。すなわち、アクセス制御条件は、エージェントクラスオブジェクト22a~22n内の置換可能なクラスデータとして管理され、動作中のエージェントクラスオブジェクト22a~22nに対してアクセス制御条件を変更することが可能となる。

【0150】このため、プログラムの追加・更新を遠隔のコンピュータから行う場合においても、プログラムコードの追加・更新のインタフェースのセキュリティを確保することが可能となり、分散オブジェクトの保守・運用に関わる管理者の負担を軽減することが可能となる。

【0151】図15は、アクセス権チェック処理を示すフローチャートである。なお、このアクセス権チェック処理は、図13のエージェントクラスオブジェクト22a~22nに対するクラスデータの追加処理、図16のクラスデータの取得処理、図19のクラスデータの実行

処理、図21のクラスデータの削除処理、図24及び図25のクラスデータの置換処理、図28のクラスデータの遠隔実行処理で呼び出される。

【0152】図15において、まず、アクセス権チェックが開始されると(ステップS41)、アクセス制御条件を記述したアクセス権チェックのクラスを共有クラスデータテーブル25から検索する(ステップS42)。そして、共有クラスデータテーブル25内にアクセス権チェックのクラスが存在するかどうかを調べ(ステップS43)、共有クラスデータテーブル25内にアクセス権チェックのクラスが存在する場合、アクセス権チェックのクラスからオブジェクトを生成し、実行する(ステップS44)。

【0153】ここで、アクセス権チェックのクラスは、処理の種類と引数、処理要求元のエージェントクラスオブジェクト22a~22nの名前“AGENT-1”~“AGENT-n”、処理要求元のユーザ名などを元にルールを記述したプログラムであり、エージェントクラスオブジェクト22a~22nに追加される他のクラスデータと同様、エージェントクラスオブジェクト22a~22nの運用中にアクセス権チェックのクラスを置き換えることを許容するものである。

【0154】このため、エージェントクラスオブジェクト22a~22nごとにアクセス制御条件を容易に変更することが可能となり、プログラムコードの追加・更新のインタフェースのセキュリティを向上させることが可能となる。

【0155】一方、共有クラスデータテーブル25内にアクセス権チェックのクラスが存在しない場合、デフォルトの条件でアクセス権のチェックを行う(ステップS

45)。

【0156】次に、アクセス権があるかどうかを判断し(ステップS46)、アクセス権がない場合は、エラーを通知して終了し(ステップS47)、アクセス権がある場合は、エージェントクラスオブジェクト22a~22nに要求があった処理を受け入れる(ステップS48)。

【0157】次に、エージェントクラスオブジェクト22a~22nからのクラスデータの取得方法について説明する。図16は、クラスデータ取得処理を示すフローチャートである。

【0158】ここで、エージェントクラスオブジェクト22a~22nがクラスデータ取得処理を行う場合、get()メソッド26cが呼び出される。図16において、まず、エージェントクラスオブジェクト22a~22nに対するアクセス権のチェックが行われ(ステップS51)、アクセス権がなければ処理が中断される。

【0159】次に、実行対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”を、get()メソッド26cの引数から取得し(ステップS52)、実行対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”を、クラス管理オブジェクトテーブル23から検索する(ステップS53)。

【0160】次に、実行対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”が、クラス管理オブジェクトテーブル23に登録されているかどうかを調べ(ステップS54)、実行対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”が、クラス管理オブジェクトテーブル23に登録されていない場合、エラーを通知して処理を中断する(ステップS55)。一方、実行対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”が、クラス管理オブジェクトテーブル23に登録されている場合、そのクラス管理オブジェクト24a~24mのクラス名リスト45からクラス名を取得し、そのクラス名のクラスデータを共有クラスデータテーブル25からコピーして(ステップS56)、そのクラスデータを戻り値として要求先に返す(ステップS57)。

【0161】次に、エージェントクラスオブジェクト22a~22nでのクラスデータの実行方法について説明する。図17は、クラスデータ実行中のエージェントクラスオブジェクトの状態を示す図である。

【0162】図17において、例えば、“function-1”という名前でクラスデータA、クラスデータB及びクラスデータCからなる機能がエージェントクラ

スオブジェクト131に追加され、クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル135に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト133が生成されている。

【0163】このクラス管理オブジェクト133のクラス名リスト(class names)には、クラス管理オブジェクト133が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、

“B”、“C”がそれぞれ登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)、参照オブジェクトリスト(referring)及び被参照オブジェクトリスト(referred)が空(null)とされている。

【0164】また、クラス管理オブジェクト133の名前として、“function-1”がクラス管理オブジェクトテーブル132に登録され、クラス管理オブジェクト132の名前“function-1”に対応して、クラス管理オブジェクト133を特定するポインタがクラス管理オブジェクトテーブル132に生成されている。

【0165】エージェントクラスオブジェクト131に対し、名前“function-1”を指定してクラスデータA、クラスデータB及びクラスデータCの実行要求が行われると、クラスデータA、クラスデータB及びクラスデータCからオブジェクトA、オブジェクトB及びオブジェクトCが生成され、スレッド134が割り当てられる。

【0166】実行中スレッドIDリスト(running threads)には、オブジェクトA、オブジェクトB及びオブジェクトCに割り当てられたスレッド134の識別情報が格納され、オブジェクトA、オブジェクトB及びオブジェクトCの実行が終了した時に、実行中スレッドIDリスト(running threads)から、オブジェクトA、オブジェクトB及びオブジェクトCに割り当てられたスレッド134の識別情報が削除される。

【0167】このため、クラス管理オブジェクト133は、実行中スレッドIDリスト(running threads)を参照することにより、自己が管理するクラスデータA、クラスデータB及びクラスデータCが使用中かどうかを容易に判定することができる。

【0168】図18は、参照関係があるクラスデータ実行中のエージェントクラスオブジェクトの状態を示す図である。図18において、例えば、“function-1”という名前で作成されたクラスデータA、クラスデータB及びクラスデータCからなる機能がエージェントクラスオブジェクト141に追加され、クラスデータA、クラス

データB及びクラスデータCが共有クラスデータテーブル147に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト143が生成されている。

【0169】このクラス管理オブジェクト143のクラス名リスト(class names)には、クラス管理オブジェクト143が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、

“B”、“C”がそれぞれ登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)及び参照オブジェクトリスト(referring)が空(null)とされている。

【0170】また、クラス管理オブジェクト143の名前として、“function-1”がクラス管理オブジェクトテーブル142に登録され、クラス管理オブジェクト143の名前“function-1”に対応して、クラス管理オブジェクト143を特定するポインタがクラス管理オブジェクトテーブル142に生成されている。

【0171】さらに、“function-4”という名前で作成されたクラスデータDからなる機能がエージェントクラスオブジェクト141に追加され、クラスデータDが共有クラスデータテーブル147に格納されるとともに、クラスデータDを管理するクラス管理オブジェクト145が生成されている。

【0172】このクラス管理オブジェクト145のクラス名リスト(class names)には、クラス管理オブジェクト145が管理するクラスデータDのクラス名“D”が登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)及び被参照オブジェクトリスト(referred)が空(null)とされている。

【0173】また、クラス管理オブジェクト145の名前として、“function-4”がクラス管理オブジェクトテーブル142に登録され、クラス管理オブジェクト145の名前“function-4”に対応して、クラス管理オブジェクト145を特定するポインタがクラス管理オブジェクトテーブル142に生成されている。

【0174】ここで、クラスデータDが、

```

class DO {
:
new AO
:
new BO
}

```

のように、例えば、クラスデータA及びクラスデータBの存在を前提としている場合、クラス管理オブジェクト145の参照オブジェクトリスト(referring)には、参照先のクラスデータA及びクラスデータBを管理しているクラス管理オブジェクト143の参照情報が格納され、クラス管理オブジェクト143の被参照オブジェクトリスト(referred)には、参照元のクラスデータDを管理しているクラス管理オブジェクト145の被参照情報が格納される。この状態で、エージェントクラスオブジェクト141に対し、名前“function-4”を指定してクラスデータDの実行要求が行われると、クラスデータDからオブジェクトDが生成され、スレッド144が割り当てられる。

【0175】クラス管理オブジェクト145の実行中スレッドIDリスト(running threads)には、オブジェクトDに割り当てられたスレッド144の識別情報が格納されるとともに、クラスデータDが参照しているクラスデータA及びクラスデータBを管理しているクラス管理オブジェクト143の実行中スレッドIDリスト(running threads)に対しても、オブジェクトDに割り当てられたスレッド144の識別情報が格納される。

【0176】そして、オブジェクトDの実行が終了した時に、クラス管理オブジェクト143及びクラス管理オブジェクト145の実行中スレッドIDリスト(running threads)から、オブジェクトDに割り当てられたスレッド144の識別情報が削除される。

【0177】このため、名前“function-4”の機能が実行されている途中に、名前“function-1”の機能の削除要求や追加要求がエージェントクラスオブジェクト141に対して行われた場合でも、クラス管理オブジェクト143は、自己の実行中スレッドIDリスト(running threads)を調べることにより、名前“function-1”の機能の削除要求や追加要求を保留することが可能となり、名前“function-4”の機能が実行されている途中に、名前“function-4”の機能の実行が不可能になることを防止することができる。

【0178】図19は、クラスデータ実行処理を示すフローチャートである。ここで、エージェントクラスオブジェクト22a～22nがクラスデータ実行処理を行う場合、call()メソッド26fが呼び出される。

【0179】図19において、まず、エージェントクラスオブジェクト22a～22nに対するアクセス権のチ

ェックが行われ(ステップS61)、アクセス権がなければ処理が中断される。

【0180】次に、実行対象のクラスデータを管理するクラス管理オブジェクト24a～24mの名前“function-1”～“function-m”をcall()メソッド26fの引数から取得し(ステップS62)、実行対象のクラスデータを管理するクラス管理オブジェクト24a～24mの名前“function-1”～“function-m”をクラス管理オブジェクトテーブル23から検索する(ステップS63)。

【0181】次に、実行対象のクラスデータを管理するクラス管理オブジェクト24a～24mの名前“function-1”～“function-m”が、クラス管理オブジェクトテーブル23に登録されているどうかを調べ(ステップS64)、実行対象のクラスデータを管理するクラス管理オブジェクト24a～24mの名前“function-1”～“function-m”が、クラス管理オブジェクトテーブル23に登録されていない場合、エラーを通知して処理を中断する(ステップS65)。

【0182】一方、実行対象のクラスデータを管理するクラス管理オブジェクト24a～24mの名前“function-1”～“function-m”が、クラス管理オブジェクトテーブル23に登録されている場合、このクラス管理オブジェクト24a～24mの実行中スレッドIDリスト44に実行中のスレッドIDを追加するとともに、このクラス管理オブジェクト24a～24mの参照オブジェクトリスト46で示されるクラス管理オブジェクト24a～24mの実行中スレッドIDリスト44に実行中のスレッドIDを追加する(ステップS66)。

【0183】次に、実行対象のクラスデータからオブジェクトを生成して処理を実行し(ステップS67)、処理が終了するのを待つ(ステップS68)。次に、処理が終わったら、このクラス管理オブジェクト24a～24mの実行中スレッドIDリスト44から実行中のスレッドIDを削除するとともに、このクラス管理オブジェクト24a～24mの参照オブジェクトリスト46で示されるクラス管理オブジェクト24a～24mの実行中スレッドIDリスト44から実行中のスレッドIDを削除する(ステップS69)。

【0184】次に、実行中スレッドIDリスト44が空かどうかを調べ(ステップS70)、実行中スレッドIDリスト44が空でない場合、実行処理を終了し(ステップS71)、実行中スレッドIDリスト44が空の場合、削除フラグ42がオンかどうかを調べる(ステップS72)。削除フラグ42がオンの場合、削除処理が予約されているため、図21の削除処理を呼び出した後(ステップS73)、実行処理を終了する(ステップS74)。

【0185】一方、削除フラグ42がオフの場合、置換クラスデータ記憶領域43が空かどうかを調べ(ステップS75)、置換クラスデータ記憶領域43が空の場合、実行処理を終了し(ステップS76)、置換クラスデータ記憶領域43が空でない場合、置換処理が予約されているため、置換クラスデータ記憶領域43に記憶されているクラスデータを引数にして、図24及び図25の置換処理を呼び出した後(ステップS77)、実行処理を終了する。

【0186】このように、動作中のエージェントクラスオブジェクト22a~22nがクラスデータの削除要求または置換要求を受け付ける場合に、クラス管理オブジェクト24a~24mが各クラスデータの使用状態を監視することにより、削除対象または置換対象のクラスデータが利用されない状態になるまで、そのクラスデータの削除または置換を延期することが可能となり、動作中のエージェントクラスオブジェクト22a~22nを途中で停止させることなく、クラスデータの削除要求や置換要求を行うことが可能となる。

【0187】なお、動作中のエージェントクラスオブジェクト22a~22nにクラスデータの削除要求または置換要求が行われた場合、削除対象または置換対象のクラスデータを使用中のオブジェクトに対して、中止要求の発行し、そのオブジェクトが応答した後にクラスデータの削除を行うようにしてもよい。

【0188】次に、エージェントクラスオブジェクト22a~22nからのクラスデータの削除方法について説明する。図20は、クラスデータ実行中に削除要求があった場合のエージェントクラスオブジェクトの状態を示す図である。

【0189】図20において、例えば、“function-1”という名前でクラスデータA、クラスデータB及びクラスデータCからなる機能がエージェントクラスオブジェクト151に追加され、クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル155に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト153が生成されている。

【0190】このクラス管理オブジェクト153のクラス名リスト(class names)には、クラス管理オブジェクト153が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、“B”、“C”がそれぞれ登録され、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)、参照オブジェクトリスト(referring)及び被参照オブジェクトリスト(referred)が空(null)とされている。

【0191】また、クラス管理オブジェクト153の名前として、“function-1”がクラス管理オブ

ジェクトテーブル152に登録され、クラス管理オブジェクト152の名前“function-1”に対応して、クラス管理オブジェクト153を特定するポインタがクラス管理オブジェクトテーブル152に生成されている。

【0192】さらに、名前“function-1”の機能が実行されている場合、クラスデータA、クラスデータB及びクラスデータCからオブジェクトA、オブジェクトB及びオブジェクトCが生成され、スレッド154が割り当てられるとともに、実行中スレッドIDリスト(running threads)には、オブジェクトA、オブジェクトB及びオブジェクトCに割り当てられたスレッド154の識別情報が格納される。

【0193】この状態で、名前“function-1”が指定され、クラスデータA、クラスデータB及びクラスデータCに対する削除要求が行われると、クラス管理オブジェクト153は、実行中スレッドIDリスト(running threads)を参照し、クラスデータA、クラスデータB及びクラスデータCの使用状況を確認する。

【0194】この場合、実行中スレッドIDリスト(running threads)には、スレッド154の識別情報が格納されているので、クラスデータA、クラスデータB及びクラスデータCは使用中であることを認識し、クラスデータA、クラスデータB及びクラスデータCに対する削除を保留するするとともに、削除フラグ(removable flag)をオンとして、クラスデータA、クラスデータB及びクラスデータCに対する削除要求があったことを記憶する。

【0195】名前“function-1”の機能の実行が終了すると、実行中スレッドIDリスト(running threads)からスレッド154の識別情報を削除し、クラス管理オブジェクト153は、削除フラグ(removable flag)の状態を参照することにより、クラスデータA、クラスデータB及びクラスデータCに対する削除要求があったかどうかを確認する。

【0196】この場合、削除フラグ(removable flag)はオンとなっているので、この時点で完全にクラスデータA、クラスデータB及びクラスデータCを削除する。

【0197】なお、強制的な削除要求であった場合は、実行中の名前“function-1”の機能を途中で停止して、クラスデータA、クラスデータB及びクラスデータCを直ちに削除するようにしてもよい。

【0198】図21は、クラスデータ削除処理を示すフローチャートである。ここで、エージェントクラスオブジェクト22a~22nがクラスデータ削除処理を行う場合、remove()メソッド26aが呼び出される。

10

20

30

40

50

【0199】図21において、まず、エージェントクラスオブジェクト22a~22nに対するアクセス権のチェックが行われ(ステップS81)、アクセス権がなければ処理が中断される。

【0200】次に、削除対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”を、remove()メソッド26aの引数から取得し(ステップS82)、削除対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”を、クラス管理オブジェクトテーブル23から検索する(ステップS83)。

【0201】次に、削除対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”が、クラス管理オブジェクトテーブル23に登録されているどうかを調べ(ステップS84)、削除対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”が、クラス管理オブジェクトテーブル23に登録されていない場合、エラーを通知して処理を中断する(ステップS85)。

【0202】一方、削除対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”が、クラス管理オブジェクトテーブル23に登録されている場合、実行中スレッドIDリスト44が空かどうかを調べ(ステップS86)、実行中スレッドIDリスト44が空でない場合、クラス管理オブジェクト24a~24mの削除フラグ42をオンにして(ステップS87)、削除処理を終了する(ステップS88)。

【0203】一方、実行中スレッドIDリスト44が空の場合、被参照オブジェクトリスト47が空かどうかを調べ(ステップS89)、被参照オブジェクトリスト47が空でなければ、エラーを通知して削除処理を中止する(ステップS90)。

【0204】一方、被参照オブジェクトリスト47が空の場合、参照オブジェクトリスト46が空かどうかを調べ(ステップS91)、参照オブジェクトリスト46が空でなければ、参照オブジェクトリスト46で示される各々のクラス管理オブジェクト24a~24mの被参照オブジェクトリスト47から、このクラス管理オブジェクト24a~24mの被参照情報を削除する(ステップS92)。

【0205】一方、参照オブジェクトリスト46が空の場合、クラス管理オブジェクト24a~24mは、クラス名リスト45を参照することにより、共有クラスデータテーブル25から削除するクラスデータを特定し、そのクラス管理オブジェクト24a~24mのクラス名リ

スト45に登録されているクラス名のクラスデータを共有クラスデータテーブルから削除する(ステップS93)。

【0206】次に、クラス管理オブジェクトテーブル23から、このクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”を削除して(ステップS94)、削除処理を終了する。

【0207】次に、エージェントクラスオブジェクト22a~22nのクラスデータの置換方法について説明する。図22は、クラスデータ実行中に置換要求があった場合のエージェントクラスオブジェクトの状態を示す図である。ここでは、名前“function-1”のクラスデータA、クラスデータB及びクラスデータCからなる機能を、名前“function-2”のクラスデータD及びクラスデータEからなる機能で置き換える例を示している。

【0208】図22において、例えば、“function-1”という名前でクラスデータA、クラスデータB及びクラスデータCからなる機能がエージェントクラスオブジェクト161に追加され、クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル165に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト163が生成されている。

【0209】このクラス管理オブジェクト163のクラス名リスト(class names)には、クラス管理オブジェクト163が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、“B”、“C”がそれぞれ登録され、削除フラグ(removeable flag)がオフに設定されるとともに、局所クラスデータテーブル(local classes)、参照オブジェクトリスト(referring)及び被参照オブジェクトリスト(referred)が空(null)とされている。また、クラス管理オブジェクト163の名前として、“function-1”がクラス管理オブジェクトテーブル162に登録され、クラス管理オブジェクト162の名前“function-1”に対応して、クラス管理オブジェクト163を特定するポインタがクラス管理オブジェクトテーブル162に生成されている。

【0210】さらに、名前“function-1”の機能が実行されている場合、クラスデータA、クラスデータB及びクラスデータCからオブジェクトA、オブジェクトB及びオブジェクトCが生成され、スレッド164が割り当てられるとともに、実行中スレッドIDリスト(running threads)には、オブジェクトA、オブジェクトB及びオブジェクトCに割り当てられたスレッド164の識別情報が格納される。

【0211】この状態で、名前“function-

10

20

30

40

50



1”が指定され、クラスデータA、クラスデータB及びクラスデータCに対する置換要求が行われると、クラス管理オブジェクト163は、実行中スレッドIDリスト (running threads) を参照し、クラスデータA、クラスデータB及びクラスデータCの使用状況を確認する。

【0212】この場合、実行中スレッドIDリスト (running threads) には、スレッド164の識別情報が格納されているので、クラスデータA、クラスデータB及びクラスデータCは使用中であることを認識し、クラスデータA、クラスデータB及びクラスデータCに対する置換を保留するするとともに、クラスデータA、クラスデータB及びクラスデータCと置換されるクラスデータD及びクラスデータEを置換クラスデータ記憶領域 (next classes) に記憶する。

【0213】名前“function-1”の機能の実行が終了すると、実行中スレッドIDリスト (running threads) からスレッド164の識別情報を削除し、クラス管理オブジェクト163は、置換クラスデータ記憶領域 (next classes) の状態を参照することにより、クラスデータA、クラスデータB及びクラスデータCに対する置換要求があったかどうかを確認する。

【0214】この場合、置換クラスデータ記憶領域 (next classes) には、クラスデータD及びクラスデータEが格納されていることから、この時点で安全にクラスデータA、クラスデータB及びクラスデータCをクラスデータD及びクラスデータEで置換する。

【0215】なお、強制的な削除要求であった場合は、実行中の名前“function-1”の機能を途中で停止して、クラスデータA、クラスデータB及びクラスデータCを直ちに置換するようにしてもよい。

【0216】図23は、クラスデータの置換後のエージェントクラスオブジェクトの状態を示す図である。図23において、クラスデータA、クラスデータB及びクラスデータCがクラスデータD及びクラスデータEで置き換えられた場合、図22のクラス管理オブジェクト163の置換クラスデータ記憶領域 (next classes) からクラスデータD及びクラスデータEが削除されるとともに、図22のクラス名リスト (class names) に登録されていたクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、

“B”、“C”が、クラスデータD及びクラスデータEのクラス名“D”、“E”で置き換えられ、図22のクラス管理オブジェクト163は、図23のクラス管理オブジェクト173に更新される。

【0217】また、図22の共有クラスデータテーブル165に格納されていたクラスデータA、クラスデータB及びクラスデータCは、クラスデータD及びクラスデータEで置き換えられ、図22の共有クラスデータテ-

ブル165は、図23の共有クラスデータテーブル174に更新される。

【0218】さらに、図22のクラス管理オブジェクトテーブル162に登録されていたクラスデータA、クラスデータB及びクラスデータCの名前“function-1”は、クラスデータD及びクラスデータEの名前“function-2”で置き換えられ、図22のクラス管理オブジェクトテーブル162は、図23のクラス管理オブジェクトテーブル172に更新される。

【0219】図24及び図25は、クラスデータ置換処理を示すフローチャートである。ここで、エージェントクラスオブジェクト22a~22nがクラスデータ削除処理を行う場合、set () メソッド26dが呼び出される。

【0220】図24において、まず、エージェントクラスオブジェクト22a~22nに対するアクセス権のチェックが行われ (ステップS101)、アクセス権がなければ処理が中断される。

【0221】次に、置換対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”及び置換する新たなクラスデータを、set () メソッド26dの引数から取得し (ステップS102)、置換対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”を、クラス管理オブジェクトテーブル23から検索する (ステップS103)。

【0222】次に、置換対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”が、クラス管理オブジェクトテーブル23に登録されているかどうかを調べ (ステップS104)、置換対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”が、クラス管理オブジェクトテーブル23に登録されていない場合、エラーを通知して処理を中断する (ステップS105)。

【0223】一方、置換対象のクラスデータを管理するクラス管理オブジェクト24a~24mの名前“function-1”~“function-m”が、クラス管理オブジェクトテーブル23に登録されている場合、実行中スレッドIDリスト44が空かどうかを調べ (ステップS106)、実行中スレッドIDリスト44が空でない場合、set () メソッド26dの引数から取得したクラスデータを置換クラスデータ記憶領域43に設定し (ステップS107)、置換処理を終了する (ステップS108)。

【0224】一方、実行中スレッドIDリスト44が空の場合、被参照オブジェクトリスト47が空かどうかを調べ (ステップS109)、被参照オブジェクトリスト

10

20

30

40

50



47が空でなければ、エラーを通知して置換処理を中止する(ステップS110)。

【0225】一方、被参照オブジェクトリスト47が空の場合、参照オブジェクトリスト46が空かどうかを調べ(ステップS111)、参照オブジェクトリスト46が空でなければ、参照オブジェクトリスト46で示される各々のクラス管理オブジェクト24a~24mの被参照オブジェクトリスト47から、このクラス管理オブジェクト24a~24mの被参照情報を削除する(ステップS112)。

【0226】一方、参照オブジェクトリスト46が空の場合、クラス管理オブジェクト24a~24mは、クラス名リスト45を参照することにより、共有クラスデータテーブル25から削除するクラスデータを特定し、そのクラス管理オブジェクト24a~24mのクラス名リスト45に登録されているクラス名のクラスデータを共有クラスデータテーブルから削除する(ステップS113)。

【0227】次に、図25に示すように、置換要求のあった新たなクラスデータを共有クラスデータテーブル125に書き込む(ステップS114)。次に、同一のクラス名が既に共有クラスデータテーブル25に存在しているかどうかを判断し(ステップS115)、クラスデータの書き込みの途中で、同一のクラス名が既に共有クラスデータテーブル25に存在していた場合、共有クラスデータテーブル25に新たに格納したクラスデータを削除し、共有クラスデータテーブル25を元の状態に戻すとともに(ステップS116)、エラーを通知して処理を終了する(ステップS117)。

【0228】一方、同一のクラス名が既に共有クラスデータテーブル25に存在していない場合、共有クラスデータテーブル25に新たに追加した全てのクラスデータのクラス名を、クラス管理オブジェクト24a~24mのクラス名リスト45に設定する(ステップS118)。

【0229】次に、共有クラスデータテーブル25に追加したクラスデータに関連するクラスを調べることにより、クラス管理オブジェクト24a~24mの参照オブジェクトリスト46に参照先情報を設定するとともに(ステップS119)、参照先のクラス管理オブジェクト24a~24mの被参照オブジェクトリスト47に参照元情報を設定して(ステップS120)、処理を終了する。

【0230】次に、エージェントクラスオブジェクト22a~22nでのクラスデータの遠隔実行方法について説明する。図26は、クラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。

【0231】図26において、例えば、“function-1”という名前でクラスデータA、クラスデータB及びクラスデータCからなる機能がエージェントクラ

スオブジェクト181に追加され、クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル185に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト183が生成されている。

【0232】このクラス管理オブジェクト183のクラス名リスト(class names)には、クラス管理オブジェクト183が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、“B”、“C”がそれぞれ登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)、実行中スレッドIDリスト(running threads)、参照オブジェクトリスト(referring)及び被参照オブジェクトリスト(referred)が空(null)とされている。

【0233】また、クラス管理オブジェクト183の名前として、“function-1”がクラス管理オブジェクトテーブル182に登録され、クラス管理オブジェクト182の名前“function-1”に対応して、クラス管理オブジェクト183を特定するポインタがクラス管理オブジェクトテーブル182に生成されている。

【0234】このエージェントクラスオブジェクト181に対し、クラスデータD、クラスデータE及びクラスデータAを引数として、eval()メソッド186が呼ばれた場合、クラス管理オブジェクト184が新たに生成される。そして、クラスデータD、クラスデータE及びクラスデータAは、このクラス管理オブジェクト184の局所クラスデータテーブル(local classes)に格納されるとともに、クラス管理オブジェクト184のクラス名リスト(class names)には、クラスデータD、クラスデータE及びクラスデータAのクラス名“D”、“E”、“A”がそれぞれ格納される。

【0235】クラス管理オブジェクト184の削除フラグ(removable flag)がオフとされ、置換クラスデータ記憶領域(next classes)、実行中スレッドIDリスト(running threads)、参照オブジェクトリスト(referring)及び被参照オブジェクトリスト(referred)が空(null)とされる。

【0236】このように、エージェントクラスオブジェクト181に対し、クラスデータD、クラスデータE及びクラスデータAからなる機能を追加する場合、クラスデータD、クラスデータE及びクラスデータAを、このクラス管理オブジェクト184の局所クラスデータテーブル(local classes)に格納することにより、共有クラスデータテーブル185にクラスデータ

10

20

30

40

50

Aが登録されている場合でも、共有クラスデータテーブル185での同一クラス名“A”の衝突を回避することが可能となり、クラスデータD、クラスデータE及びクラスデータAからなる機能を実行させることが可能となる。

【0237】このクラスデータD、クラスデータE及びクラスデータAを、クラス管理オブジェクト184の局所クラスデータテーブル(local classes)に格納する方法は、クラスデータD、クラスデータE及びクラスデータAからなる機能の使用頻度が少ない場合や、実行されるクラスデータが頻繁に変更される場合に有効である。

【0238】図27は、シリアル化データの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。図27において、エージェントクラスオブジェクト191の共有クラスデータテーブル192にはクラスデータDが格納され、クラスデータDはその属性として、クラスStringのオブジェクトとクラスIntegerのオブジェクトを持つように定義されている。また、エージェントクラスオブジェクト191には、クラスデータDのインスタンス193が生成されている。

【0239】このエージェントクラスオブジェクト191が、クラスデータD及びクラスデータDのインスタンス193を引数として、エージェントクラスオブジェクト194のeval()メソッド195を呼び出す場合、エージェントクラスオブジェクト191は、クラスデータDのインスタンス193をシリアル化化する。

【0240】そして、クラスデータD及びインスタンス193のシリアル化データをeval()メソッド195の引数に設定し、「“D” {Strings; Integer;}、 “D”, String, “19:50AM”, Integer, “19:50AM”」を送る。

【0241】エージェントクラスオブジェクト194は、eval()メソッド195が呼び出されると、クラス管理オブジェクト196を生成し、クラスデータDの内容 {Strings; Integer;} をクラス管理オブジェクト196の局所クラスデータテーブル(local classes)に格納するとともに、クラス管理オブジェクト196のクラス名リスト(class names)にクラスデータDのクラス名 “D” を格納する。

【0242】また、クラス管理オブジェクト196の削除フラグ(removable flag)がオフとされ、置換クラスデータ記憶領域(next classes)、実行中スレッドIDリスト(running threads)、参照オブジェクトリスト(referring)及び被参照オブジェクトリスト(referred)が空(null)とされる。

【0243】さらに、クラスデータD及びインスタンス193のシリアル化データに基づいて、エージェントクラスオブジェクト191のインスタンス193を復元したインスタンス197を生成し、クラスデータD及びインスタンス197の実行を行う。

【0244】このように、エージェントクラスオブジェクト194は、クラスデータ及びそのクラスデータから生成されたオブジェクトのシリアル化データを他のエージェントクラスオブジェクト191から受信した場合、受信したクラスデータから新たにオブジェクトを生成するのではなく、受信したクラスデータとシリアル化データから送信元のオブジェクトの状態を復元して遠隔実行を行うことにより、他のエージェントクラスオブジェクト191のインスタンス193の実行が可能となる。

【0245】図28は、遠隔実行処理を示すフローチャートである。エージェントクラスオブジェクト22a～22nがクラスデータ遠隔実行処理を行う場合、eval()メソッド26eが呼び出される。

【0246】図28において、まず、エージェントクラスオブジェクト22a～22nに対するアクセス権のチェックが行われ(ステップS130)、アクセス権がなければ処理が中断される。

【0247】次に、eval()メソッド26eの引数からクラスデータとシリアル化されたオブジェクトデータを取得し(ステップS131)、このクラスデータを管理するクラス管理オブジェクト24a～24mを新たに生成するとともに、このクラス管理オブジェクト24a～24mの局所クラスデータテーブル41に引数のクラスデータを設定する(ステップS132)。

【0248】次に、局所クラスデータテーブル41に追加したクラスデータに関連するクラスを調べることにより、そのクラス管理オブジェクト24a～24mの参照オブジェクトリスト46に参照先情報を設定するとともに(ステップS133)、参照先のクラス管理オブジェクト24a～24mの被参照オブジェクトリスト47に参照元情報を設定する(ステップS134)。

【0249】次に、このクラス管理オブジェクト24a～24mの実行中スレッドIDリスト44に実行中のスレッドIDを追加するとともに、このクラス管理オブジェクト24a～24mの参照オブジェクトリスト46で示されるクラス管理オブジェクト24a～24mの実行中スレッドIDリスト44に実行中のスレッドIDを追加する(ステップS135)。

【0250】次に、eval()メソッド26eの引数にシリアル化されたオブジェクトデータがあるかどうかを調べ(ステップS136)、eval()メソッド26eの引数にシリアル化されたオブジェクトデータがない場合、実行対象のクラスデータからオブジェクトを生成して処理を実行し(ステップS137)、そ

のオブジェクトの処理が終了するのを待つ（ステップS139）。

【0251】一方、eval () メソッド26eの引数にシリアル化されたオブジェクトデータがある場合、シリアル化されたオブジェクトデータ及び局所クラスデータテーブル41のクラスデータを用いてそのオブジェクトを復元し、そのオブジェクトの処理を実行する（ステップS138）。

【0252】次に、処理が終わったら、このクラス管理オブジェクト24a~24mの実行中スレッドIDリスト44から実行中のスレッドIDを削除するとともに、このクラス管理オブジェクト24a~24mの参照オブジェクトリスト46で示されるクラス管理オブジェクト24a~24mの実行中スレッドIDリスト44から実行中のスレッドIDを削除する（ステップS140）。

【0253】次に、参照オブジェクトリスト46で示される各々のクラス管理オブジェクト24a~24mの被参照オブジェクトリスト47から、このクラス管理オブジェクト24a~24mの被参照情報を削除し（ステップS141）、実行対象のクラスデータを管理していたクラス管理オブジェクト24a~24mを削除する（ステップS142）。

【0254】図29は、参照関係を有するクラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。図29において、例えば、“function-1”という名前でクラスデータA、クラスデータB及びクラスデータCからなる機能がエージェントクラスオブジェクト201に追加され、クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル206に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト203が生成されている。

【0255】このクラス管理オブジェクト203のクラス名リスト(class names)には、クラス管理オブジェクト203が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、“B”、“C”がそれぞれ登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)、実行中スレッドIDリスト(running threads)及び参照オブジェクトリスト(referring)が空(null)とされている。

【0256】また、クラス管理オブジェクト203の名前として、“function-1”がクラス管理オブジェクトテーブル202に登録され、クラス管理オブジェクト203の名前“function-1”に対応して、クラス管理オブジェクト203を特定するポインタがクラス管理オブジェクトテーブル202に生成されている。

【0257】さらに、“function-4”という名前前でクラスデータDからなる機能がエージェントクラスオブジェクト201に追加され、クラスデータDが共有クラスデータテーブル206に格納されるとともに、クラスデータDを管理するクラス管理オブジェクト204が生成されている。

【0258】このクラス管理オブジェクト204のクラス名リスト(class names)には、クラス管理オブジェクト204が管理するクラスデータDのクラス名“D”が登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)、実行中スレッドIDリスト(running threads)及び被参照オブジェクトリスト(referrde)が空(null)とされている。

【0259】また、クラス管理オブジェクト204の名前として、“function-4”がクラス管理オブジェクトテーブル202に登録され、クラス管理オブジェクト204の名前“function-4”に対応して、クラス管理オブジェクト204を特定するポインタがクラス管理オブジェクトテーブル202に生成されている。

【0260】ここで、クラスデータDが、

```
class D0 {
    :
    new A0
    :
    new B0
}
```

のように、例えば、クラスデータA及びクラスデータBの存在を前提としている場合、クラス管理オブジェクト204の参照オブジェクトリスト(referring)には、参照先のクラスデータA及びクラスデータBを管理しているクラス管理オブジェクト203の参照情報が格納され、クラス管理オブジェクト203の被参照オブジェクトリスト(referred)には、参照元のクラスデータDを管理しているクラス管理オブジェクト204の被参照情報が格納される。

【0261】なお、クラスの呼び出し関係は、例えば、Java言語の場合、クラスのメソッドであるgetDeclaredClasses ()を用いることにより調べることができる。

【0262】この状態で、エージェントクラスオブジェクト201が、名前“function-4”を指定してクラスデータDの遠隔実行の要求を行うと、スレッド205が生成され、クラスデータDを送るとともに、クラス管理オブジェクト204の参照オブジェクトリスト(referring)により示されるクラス管理オブジェクト203が管理しているクラスデータA、クラス

データB及びクラスデータCも一緒に送る。

【0263】ここで、クラス管理オブジェクト203が管理しているクラスデータA、クラスデータB及びクラスデータCと、クラス管理オブジェクト204が管理しているクラスデータDとは、エージェントクラスオブジェクト201の共有クラスデータテーブル206に格納され、同一のエージェントクラスオブジェクト201で管理されていることから、クラスデータDが参照しているクラスデータA及びクラスデータBを容易に取得することが可能である。

【0264】このように、遠隔実行するクラスとそのクラスの実行時に必要となる他のクラスとの呼出し関係をクラスデータの転送前に調べ、遠隔実行するクラスデータを送るときに、そのクラスデータの実行時に必要となる他のクラスデータも一緒に送ることにより、転送先での処理の負担を軽減することができ、特に、分散システムなどのようにクラスデータが分散して格納されている場合に有効である。

【0265】図30は、重複関係を有するクラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。図30において、例えば、“function-5”という名前前でクラスデータA、クラスデータD及びクラスデータEからなる機能がエージェントクラスオブジェクト211に追加され、クラスデータA、クラスデータD及びクラスデータEが共有クラスデータテーブル214に格納されるとともに、クラスデータA、クラスデータD及びクラスデータEを管理するクラス管理オブジェクト213が生成されている。

【0266】このクラス管理オブジェクト213のクラス名リスト(class names)には、クラス管理オブジェクト213が管理するクラスデータA、クラスデータD及びクラスデータEのクラス名“A”、“D”、“E”がそれぞれ登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)、実行中スレッドIDリスト(running threads)、参照オブジェクトリスト(referring)及び被参照オブジェクトリスト(referred)が空(null)とされている。

【0267】また、クラス管理オブジェクト213の名前として、“function-5”がクラス管理オブジェクトテーブル212に登録され、クラス管理オブジェクト213の名前“function-5”に対応して、クラス管理オブジェクト213を特定するポインタがクラス管理オブジェクトテーブル212に生成されている。

【0268】また、例えば、“function-1”という名前前でクラスデータA、クラスデータB及びクラスデータCからなる機能がエージェントクラスオブジェ

クト215に追加され、クラスデータA、クラスデータB及びクラスデータCが共有クラスデータテーブル218に格納されるとともに、クラスデータA、クラスデータB及びクラスデータCを管理するクラス管理オブジェクト217が生成されている。

【0269】このクラス管理オブジェクト217のクラス名リスト(class names)には、クラス管理オブジェクト217が管理するクラスデータA、クラスデータB及びクラスデータCのクラス名“A”、“B”、“C”がそれぞれ登録され、削除フラグ(removable flag)がオフとされ、局所クラスデータテーブル(local classes)、置換クラスデータ記憶領域(next classes)、実行中スレッドIDリスト(running threads)、参照オブジェクトリスト(referring)及び被参照オブジェクトリスト(referred)が空(null)とされている。

【0270】また、クラス管理オブジェクト217の名前として、“function-1”がクラス管理オブジェクトテーブル216に登録され、クラス管理オブジェクト217の名前“function-1”に対応して、クラス管理オブジェクト217を特定するポインタがクラス管理オブジェクトテーブル216に生成されている。

【0271】エージェントクラスオブジェクト211が、名前“function-5”の機能の遠隔実行をエージェントクラスオブジェクト215で行う場合、名前“function-5”の機能を構成するクラスデータD、クラスデータE及びクラスデータAが、エージェントクラスオブジェクト215の共有クラスデータテーブル218に存在するかどうかを問い合わせる。

【0272】エージェントクラスオブジェクト215は、クラスデータD、クラスデータE及びクラスデータAが共有クラスデータテーブル218に存在するかどうかを調べ、問い合わせを受けたクラスデータD、クラスデータE及びクラスデータAのうち、クラスデータAが共有クラスデータテーブル218に存在することを認識する。そして、クラスデータAの存在をエージェントクラスオブジェクト215に回答する。

【0273】クラスデータAの存在を知らされたエージェントクラスオブジェクト211は、エージェントクラスオブジェクト215に転送するクラスデータD、クラスデータE及びクラスデータAからクラスデータAを除き、クラスデータD及びクラスデータEだけをエージェントクラスオブジェクト215に送る。

【0274】エージェントクラスオブジェクト215は、エージェントクラスオブジェクト215から転送されたクラスデータD及びクラスデータEを用いるとともに、自己の共有クラスデータテーブル218に格納されているクラスデータAを用いることにより、名前“f u

10

20

30

40

50

nction-5”の機能を実行する。

【0275】このことにより、遠隔実行する際に、転送先の共有クラスデータテーブルに格納されているクラスデータを送る必要がなくなり、クラスデータの転送量を削減することができる。

【0276】次に、エージェントクラスオブジェクト22a~22nによるクラスデータの検索方法について説明する。クラス管理オブジェクト24a~24mはクラス検索機能48を備えており、エージェントクラスオブジェクト22a~22nに追加されたクラスデータからオブジェクトを生成するときや、オブジェクトのメソッド内部で宣言されているクラスデータからオブジェクトを生成する時には、クラス管理オブジェクト24a~24mのloadClass()メソッドを呼び出す。

【0277】図31は、クラス検索処理を示すフローチャートである。図31において、まず、loadClass()メソッドの引数から検索対象のクラスデータのクラス名を取得し(ステップS151)、局所クラスデータテーブル41を検索する(ステップS152)。そして、局所クラスデータテーブル41の検索結果を調べ(ステップS153)、検索対象のクラスデータが見つかった場合、そのクラスデータを戻り値として返す(ステップS158)。

【0278】一方、検索対象のクラスデータが局所クラスデータテーブル41に見つからなかった場合、共有クラスデータテーブル25を検索する(ステップS154)。そして、共有クラスデータテーブル25の検索結果を調べ(ステップS155)、検索対象のクラスデータが見つかった場合、そのクラスデータを戻り値として返す(ステップS158)。

【0279】一方、検索対象のクラスデータが共有クラスデータテーブル25に見つからなかった場合、エージェントクラスライブラリ31及びシステムクラスライブラリ32を検索する(ステップS156)。そして、エージェントクラスライブラリ31及びシステムクラスライブラリ32の検索結果を調べ(ステップS157)、検索対象のクラスデータが見つかった場合、そのクラスデータを戻り値として返す(ステップS158)。

【0280】一方、検索対象のクラスデータがエージェントクラスライブラリ31及びシステムクラスライブラリ32に見つからなかった場合、エラーを通知して終了する(ステップS159)。

【0281】以上説明したように、本発明の実施例によれば、分散オブジェクトにクラスデータの置換機能と遠隔実行機能を付加したエージェントクラスを用意することで、柔軟な分散オブジェクトシステムの運用が可能となり、開発者にクラス管理の知識がなくても容易に安全にこれらの機能を利用することが可能となる。

【0282】以上、本発明の実施例について説明したが、本発明は上述した実施例に限定されることなく、本

発明の技術的思想の範囲内で他の様々の変更が可能である。

【0283】

【発明の効果】以上説明したように、本発明によれば、プログラムの使用状況を追加されたクラスデータごとに監視することにより、プログラムが動作している場合にどのクラスデータが使用されているかを判別することが可能となることから、そのプログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となる。

【0284】また、本発明の一態様によれば、動作中のプログラムに対して削除や置換などの処理要求が行われた場合においても、そのプログラムの動作を停止させることなく、そのプログラムに対する削除や置換などの処理を行わせることが可能となり、プログラムのカスタマイズ時におけるプログラムの負担を軽減することが可能となる。

【0285】また、本発明の一態様によれば、クラスデータの使用状況を監視する監視オブジェクトを監視対象のオブジェクト内に設けることにより、監視対象となるプログラム自体にそのプログラムを監視する機能を容易に付加することが可能となり、プログラマにクラス管理の知識がなくても、プログラムの使用状況の監視を容易に行うことが可能となるとともに、そのプログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となる。

【0286】また、本発明の一態様によれば、クラスデータの追加要求が行われるごとに監視オブジェクトを生成することにより、プログラムに新たに追加された機能ごとにプログラムの監視を行うことが可能となり、プログラムが動作している場合にプログラムのどの機能が使用されているのかを判別することが可能となることから、プログラムの一部の機能を特定するだけで、そのプログラムの機能の削除や置換をそのプログラムの動作中に行うことが可能となる。

【0287】また、本発明の一態様によれば、クラス管理オブジェクトの識別情報を記憶することにより、クラス管理オブジェクトに与えられた名前などの識別情報を指定するだけで、プログラムコードの一部の削除や置換をそのプログラムの動作中に行うことが可能となり、プログラムをカスタマイズする際のプログラマの負担を軽減することが可能となる。

【0288】また、本発明の一態様によれば、クラス管理オブジェクトに固有のクラスデータを格納する局所クラスデータテーブルを設けることにより、他のオブジェクトから送られてきたクラスデータとクラス管理オブジェクトが既に管理しているクラスデータとの間に衝突が発生した場合においても、他のオブジェクトから送られてきたクラスデータをクラス管理オブジェクトが既に管理しているクラスデータと独立に保有することが可能と

なり、エージェントクラスオブジェクトは、他のオブジェクトから送られてきたクラスデータを遠隔実行することが可能となる。

【0289】また、本発明の一態様によれば、クラスデータに対する削除要求を記憶しておくことにより、削除要求されたクラスデータが使用中の場合においても、そのクラスデータに対する削除要求を受け入れることが可能となることから、クラスデータに対する削除要求が拒否されたために、そのクラスデータに対する削除要求を改めて行う必要がなくなり、プログラマがクラスデータを削除する際の負担を軽減することが可能となる。

【0290】また、本発明の一態様によれば、置換要求時の新たなクラスデータを記憶しておくことにより、置換要求されたクラスデータが使用中の場合においても、そのクラスデータに対する置換要求を受け入れることが可能となることから、クラスデータに対する置換要求が拒否されたために、そのクラスデータに対する置換要求を改めて行う必要がなくなり、プログラマがクラスデータを置換する際の負担を軽減することが可能となる。

【0291】また、本発明の一態様によれば、クラスデータを実行しているスレッドを記憶することにより、実行中スレッドリストを参照するだけで、実行されているクラスデータが存在するかどうかを判別することができ、クラスデータの使用状況を容易に把握することが可能となる。

【0292】また、本発明の一態様によれば、追加されたクラスデータに対応するクラス名を記憶することにより、クラス管理オブジェクトが管理しているクラスデータを容易に把握することが可能となり、追加されたクラスデータを単位としてクラスデータに対する処理を行うことが可能となる。

【0293】また、本発明の一態様によれば、クラスデータの参照関係をクラス管理オブジェクトに記憶することにより、クラスデータの参照関係を考慮しながらクラスデータに対する処理を行うことが可能となり、プログラマがクラスデータの参照関係を認識する必要がなくなることから、プログラムのカスタマイズを容易に行うことが可能となる。

【0294】また、本発明の一態様によれば、他のエージェントクラスオブジェクトからのアクセス権をチェックするアクセス制御クラスを設けることにより、エージェントクラスオブジェクトは、他のエージェントクラスオブジェクトからのアクセスがあった場合、アクセス制御クラスに基づいてアクセス制御オブジェクトを生成することが可能となることから、エージェントクラスオブジェクトからのアクセスがある度に、アクセス権のチェックを行うことが可能となるとともに、エージェントクラスオブジェクトごとにアクセス条件を容易に変更することが可能となり、プログラムコードの追加・更新のインターフェイスのセキュリティを向上させることが可能

となる。

【0295】また、本発明の一態様によれば、クラスデータのクラス名の重複を排除することにより、共有クラスデータテーブル内で同一のクラスの衝突が発生することを防止することができる。

【0296】また、本発明の一態様によれば、参照元のクラスデータの実行中に参照先のクラスデータの削除要求や置換要求がなされた場合、参照元のクラスデータの実行が終了するまで参照先のクラスデータの削除や置換を一時的に保留しておくことにより、クラスデータの実行が途中で不可能になることを防止することが可能となる。

【0297】また、本発明の一態様によれば、削除要求や置換要求があったプログラムの一部の機能について、そのプログラムの一部の機能の実行が終了した後に、そのプログラムの一部の機能を削除したり、置換したりすることにより、プログラムの一部の機能を削除または置換する際に、実行中のプログラムを途中で停止させたり、そのプログラムを再起動させたりする手間を省くことが可能となる。

【0298】また、本発明の一態様によれば、他のエージェントクラスオブジェクトから送られたクラスデータを局所的に格納することにより、クラス管理オブジェクトに対応して既に格納されているクラスデータとの衝突を回避することが可能となる、エージェントクラスオブジェクトによる遠隔実行を円滑に行うことが可能となる。

【0299】また、本発明の一態様によれば、クラスデータ及びクラスデータから生成されたオブジェクトのシリアル化データに基づいて、そのクラスデータから生成されたオブジェクトを復元するようにしている。

【0300】このことにより、クラスデータから生成されたオブジェクトをシリアル化して送信し、シリアル化データを受信先で復元することにより、オブジェクトの遠隔実行を行うことが可能となる。

【0301】また、本発明の一態様によれば、実行対象のクラスデータが参照しているクラスデータを送信することにより、参照関係を有しているクラスデータについても、遠隔実行を行うことが可能となる。

【0302】また、本発明の一態様によれば、送信先のエージェントクラスオブジェクトに存在しないクラスデータだけを送信することにより、遠隔実行を行う際の通信量を減らすことが可能となり、遠隔実行の速度を上げることが可能となる。

【図面の簡単な説明】

【図1】本発明の一実施例に係わる情報処理装置の機能的な構成を示すブロック図である。

【図2】図1のクラスデータ処理手段の構成例を示すブロック図である。

【図3】本発明の一実施例に係わるコンピュータの機能

的な構成を示すブロック図である。

【図4】図3のクラス管理オブジェクトの構成例を示す図である。

【図5】エージェントクラスの構成例を示す図である。

【図6】クラスデータ追加時のクラス管理オブジェクトの生成方法を示す図である。

【図7】テンプレートのコード化の例を示す図である。

【図8】図3のコンピュータのシステム構成例を示す図である。

【図9】図3の通信デーモンの動作を示すフローチャートである。

【図10】図3の通信スレッドの動作を示すフローチャートである。

【図11】クラスデータをエージェントクラスオブジェクトに追加した状態を示す図である。

【図12】参照関係のあるクラスデータをエージェントクラスオブジェクトに追加した状態を示す図である。

【図13】重複するクラスデータをエージェントクラスオブジェクトから削除する方法を示す図である。

【図14】クラスデータ追加処理を示すフローチャートである。

【図15】アクセス権チェック処理を示すフローチャートである。

【図16】クラスデータ取得処理を示すフローチャートである。

【図17】クラスデータ実行中のエージェントクラスオブジェクトの状態を示す図である。

【図18】参照関係があるクラスデータ実行中のエージェントクラスオブジェクトの状態を示す図である。

【図19】クラスデータ実行処理を示すフローチャートである。

【図20】クラスデータ実行中に削除要求があった場合のエージェントクラスオブジェクトの状態を示す図である。

【図21】クラスデータ削除処理を示すフローチャートである。

【図22】クラスデータ実行中に置換要求があった場合のエージェントクラスオブジェクトの状態を示す図である。

【図23】クラスデータの置換後のエージェントクラスオブジェクトの状態を示す図である。

【図24】クラスデータ置換処理を示すフローチャートである。

【図25】クラスデータ置換処理を示すフローチャート(続き)である。

【図26】クラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。

【図27】シリアルライズデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。

【図28】遠隔実行処理を示すフローチャートである。

【図29】参照関係を有するクラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。

【図30】重複関係を有するクラスデータの遠隔実行時のエージェントクラスオブジェクトの状態を示す図である。

【図31】クラス検索処理を示すフローチャートである。

【符号の説明】

- 1 クラスデータ記憶手段
- 2 クラスデータ監視手段
- 3 クラスデータ処理手段
  - 11 受付手段
  - 12 判定手段
  - 13 延期手段
- 21 コンピュータ
- 22 a~22 n, 101, 111, 121, 131, 141, 151, 161, 171, 181, 191, 194, 201, 211, 215, 301 エージェントクラスオブジェクト
- 23, 53, 102, 112, 122, 132, 142, 152, 162, 172, 182, 202, 212, 216 クラス管理オブジェクトテーブル
- 24 a~24 m, 54, 103, 113, 114, 123, 133, 143, 145, 153, 163, 173, 183, 184, 196, 203, 204, 213, 217, 303, 304, 305 クラス管理オブジェクト
- 25, 55, 104, 115, 124, 135, 147, 155, 165, 174, 185, 192, 206, 214, 218, 302 共有クラスデータテーブル
- 26 a~26 f, 56, 125, 186, 195 外部公開メソッド
- 27 a~27 k, 134, 144, 146, 154, 164, 205 通信スレッド
- 28 通信デーモン
- 29 ネットワークソケットAPI
- 30 オペレーティングシステム
- 31 エージェントクラスライブラリ
- 32 システムクラスライブラリ
- 41 局所クラスデータテーブル
- 42 削除フラグ
- 43 置換クラスデータ格納部
- 44 実行中スレッドIDリスト
- 45 クラス名リスト
- 46 参照オブジェクトリスト
- 47 被参照オブジェクトリスト
- 51 クラスライブラリ
- 52 基本クラス



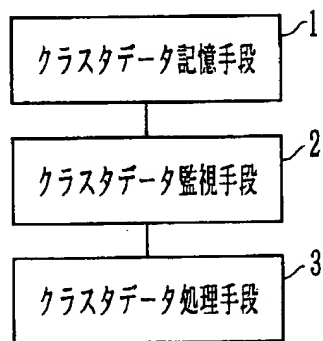
57 サブクラス  
 58 a～58 j、193、197 インスタンス  
 61 CPU  
 62 RAM  
 63 ROM  
 64 通信インターフェイス  
 65 プリンタ  
 66 ディスプレイ  
 67 キーボード  
 68 マウス

【図1】

69 ドライバ  
 70 ハードディスク  
 71 フロッピーディスク  
 72 磁気テープ  
 73 光ディスク  
 74 ICメモ리카ード  
 81 初期メニュー  
 82 ホップアップメニュー  
 83 フィールド欄  
 10 84 内容欄

【図2】

本発明の一実施例に係わる  
 情報処理装置の機能的な構成を示すブロック図



【図9】

\*1のクラスデータ処理手段の  
 構成例を示すブロック図

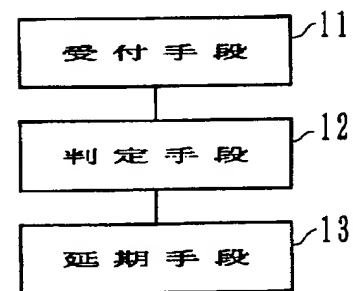
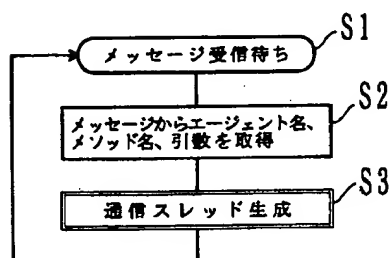
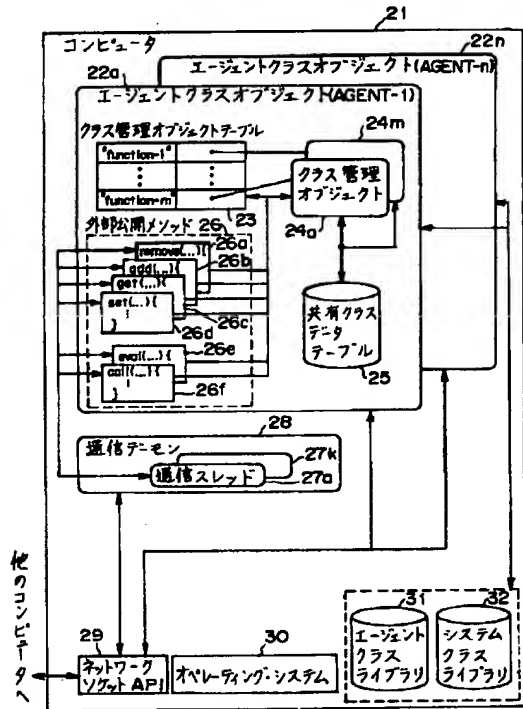


図3の通信デモンの動作を示すフローチャート



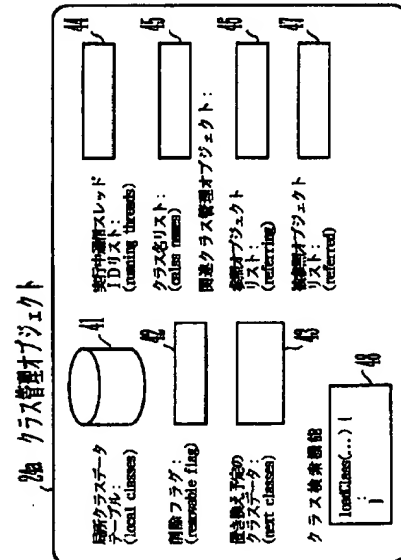
【図3】

本発明の一実施例に係るコンピュータの  
機能的な構成を示すブロック図



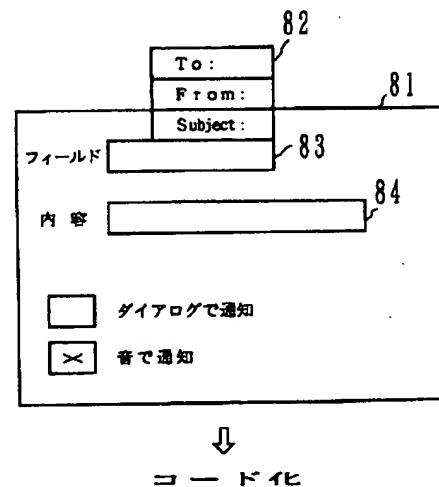
【図4】

図3のクラス管理オブジェクトの構成例を示す図



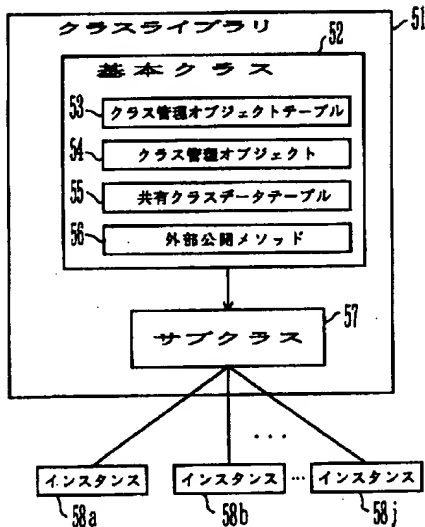
【図7】

テンプレートのコード化の例を示す図

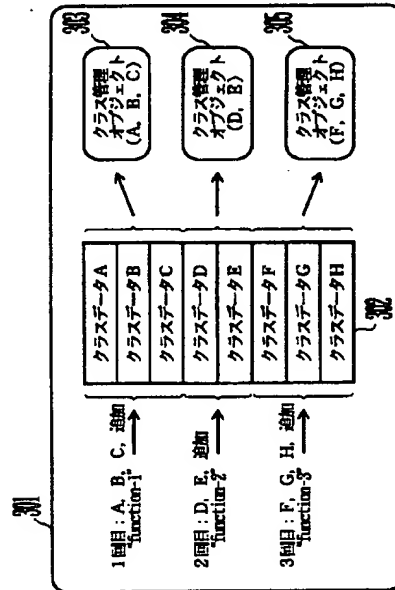


【図5】

エージェントクラスの構成例を示す図

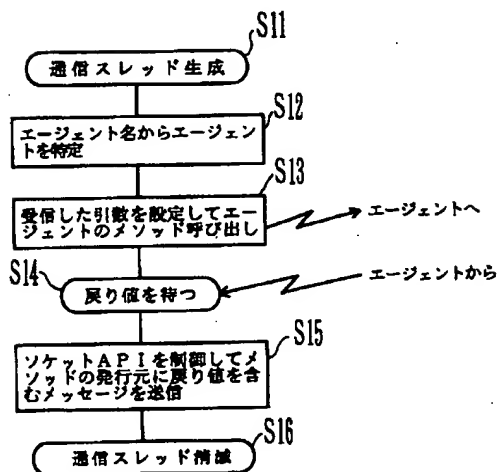


【図6】

クラスデータ追加時の  
クラス管理オブジェクトの生成方法を示す図

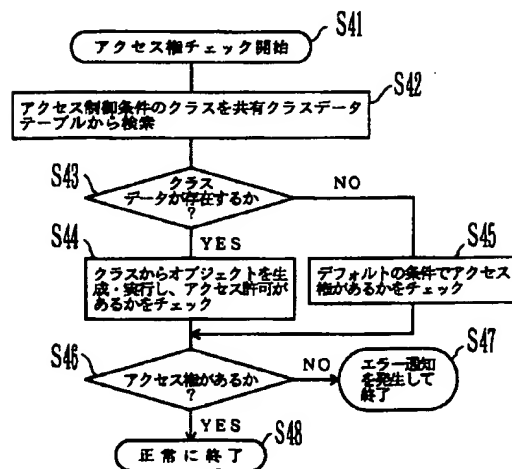
【図10】

図3の通信スレッドの動作を示すフローチャート



【図15】

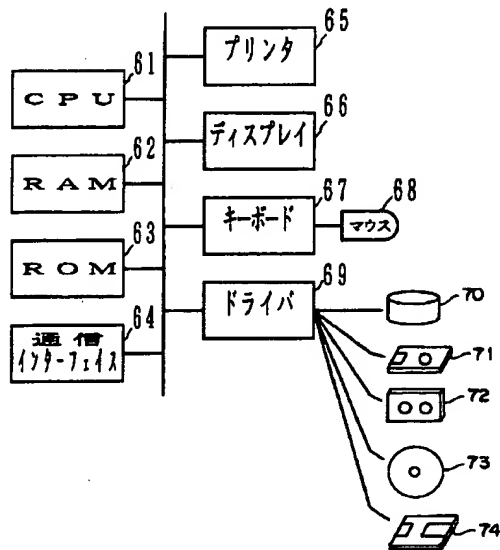
アクセス権チェック処理を示すフローチャート



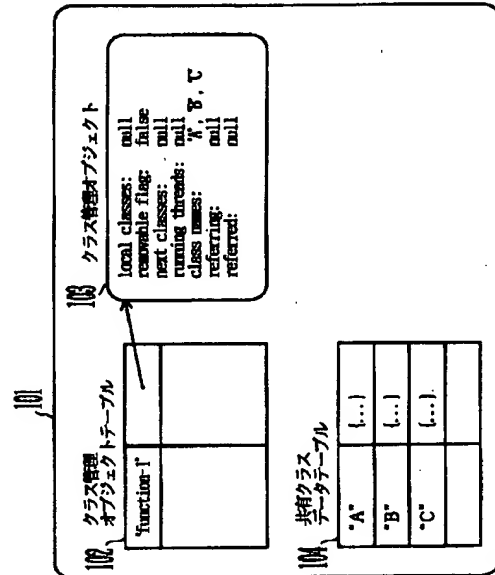
【図8】

【図11】

図3のコンピュータのシステム構成例を示す図

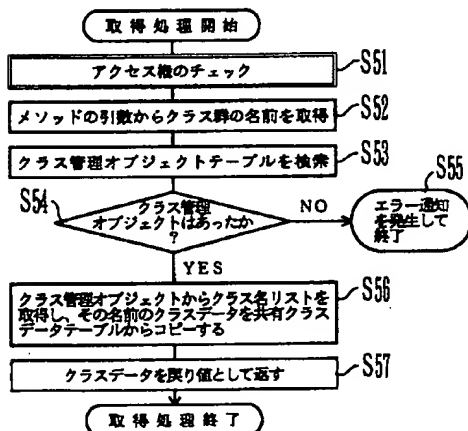


クラスデータをエージェントクラスオブジェクトに追加した状態を示す図



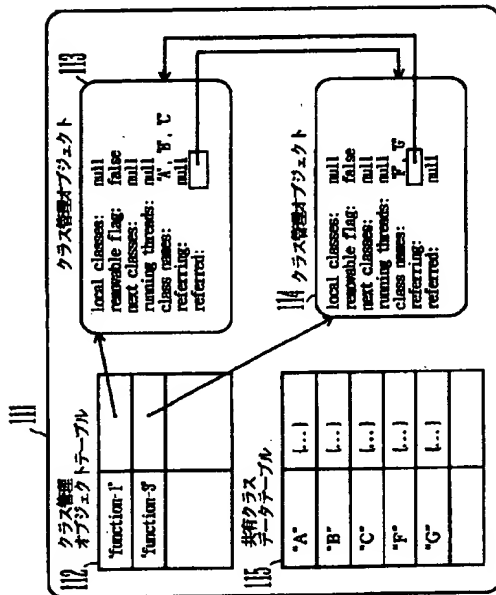
【図16】

クラスデータ取得処理を示すフローチャート



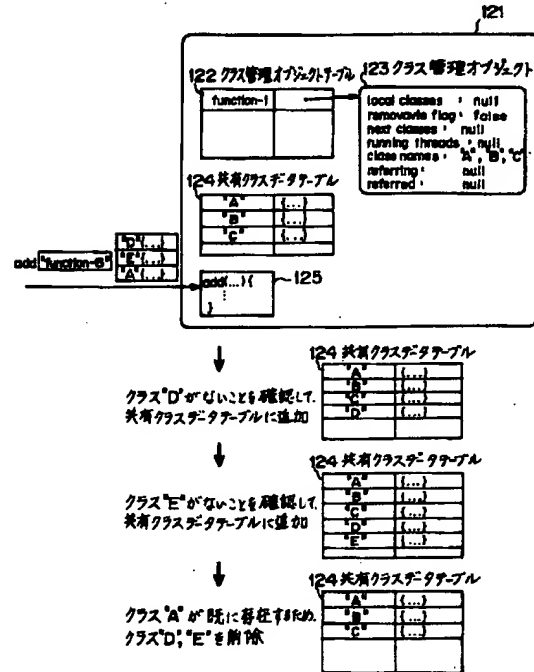
【図12】

参照関係のあるクラスデータをエージェントクラス  
オブジェクトに追加した状態を示す図



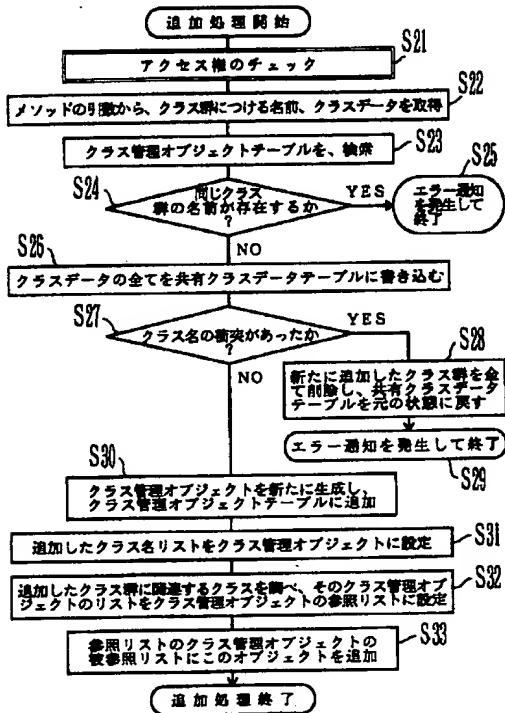
【図13】

重複するクラスデータをエージェントクラスオブジェクトから  
削除する方法を示す図



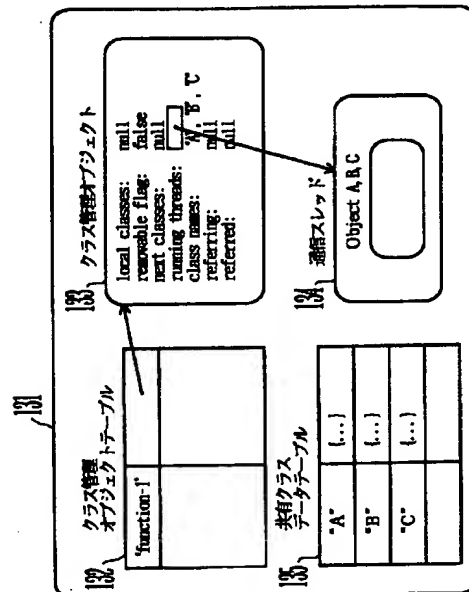
【図14】

クラスデータ追加処理を示すフローチャート



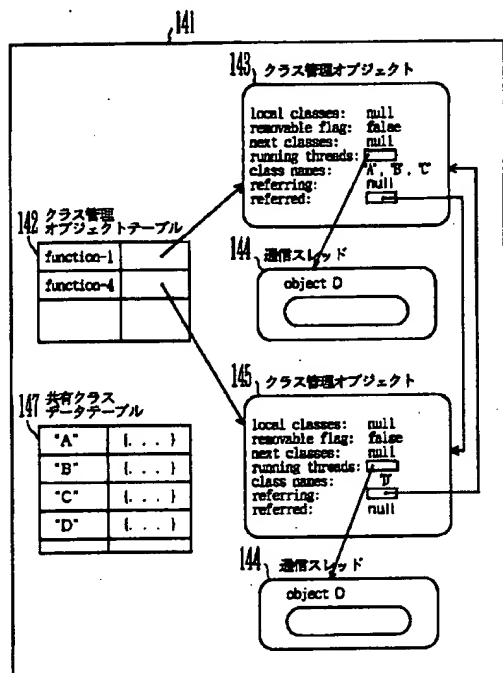
【図17】

クラスデータ実行中のエージェントクラスオブジェクトの状態表示図



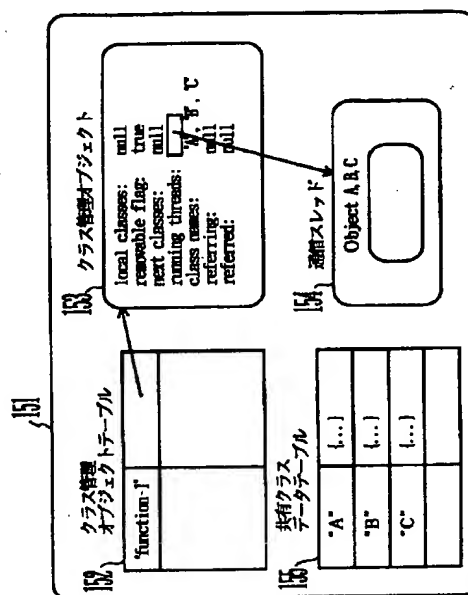
【図18】

参照関係があるクラスデータ実行中の  
エージェントクラスオブジェクトの状態を示す図



【図20】

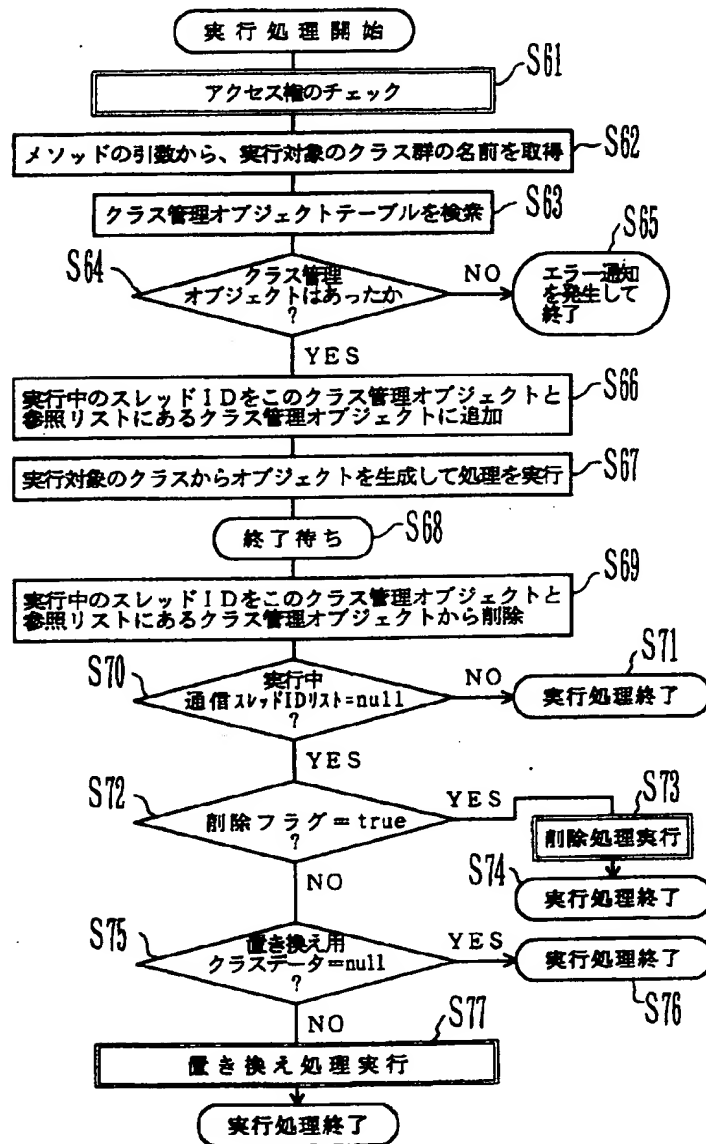
クラスデータ実行中に削除要求があった場合の  
エージェントクラスオブジェクトの状態を示す図





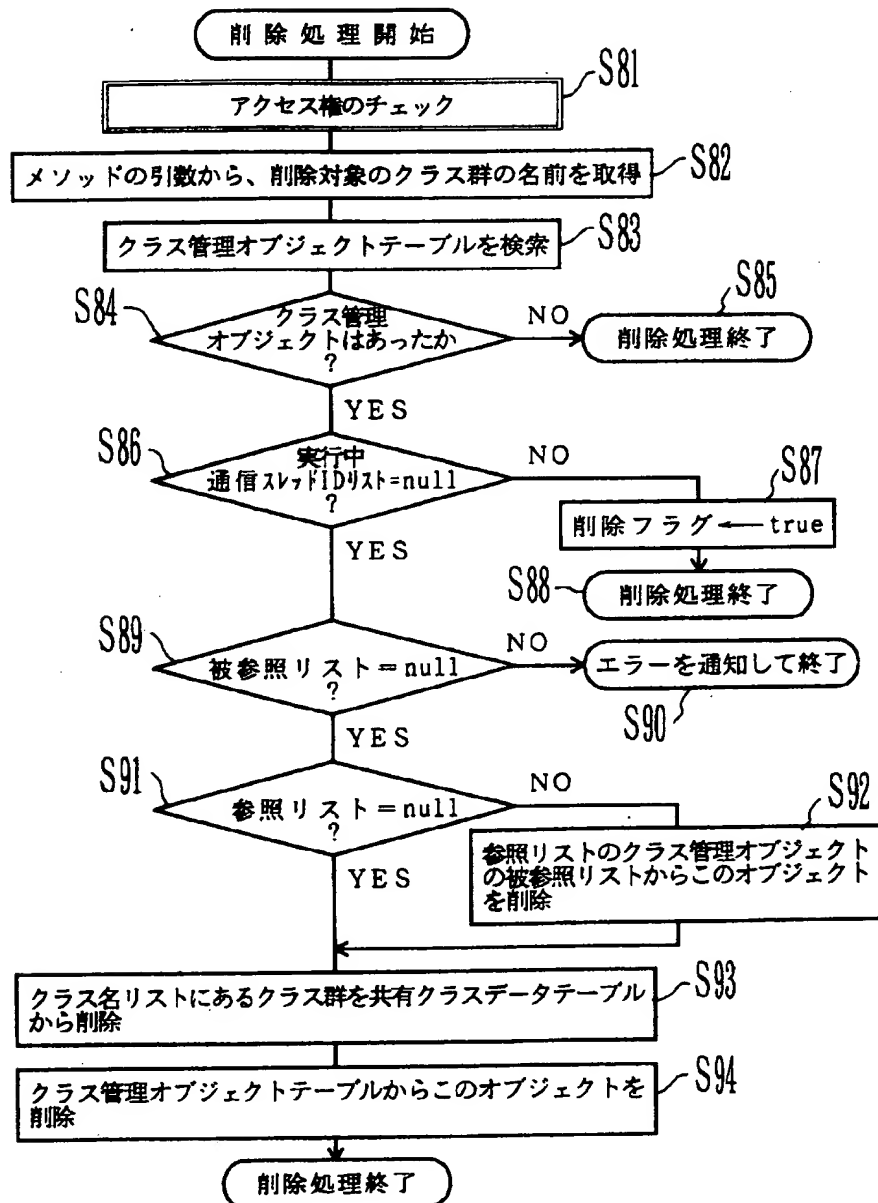
【図19】

## クラスデータ実行処理を示すフローチャート



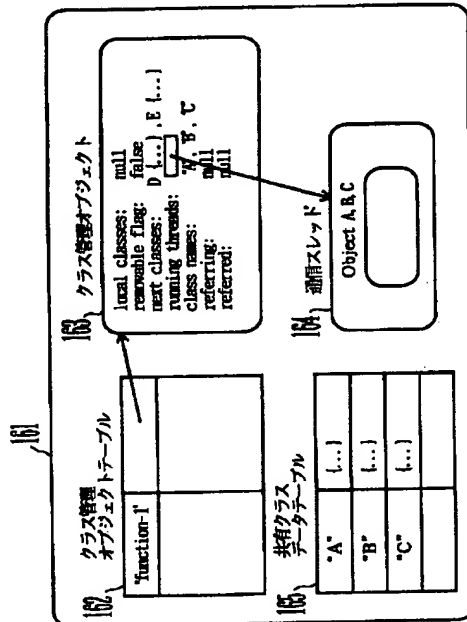
【図21】

## クラスデータ削除処理を示すフローチャート



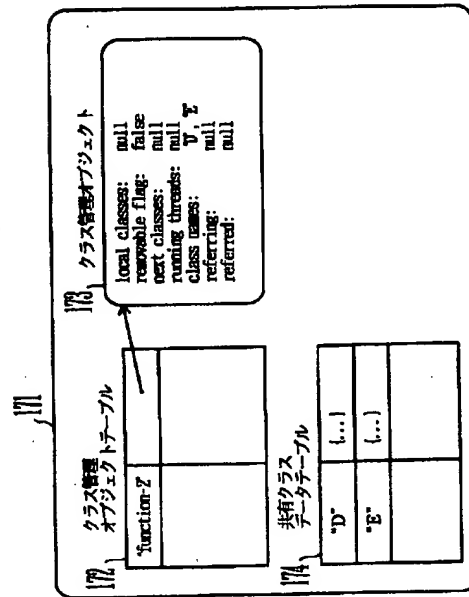
【図22】

クラスデータ実行中に置換要求があった場合の  
エージェントクラスオブジェクトの状態を示す図



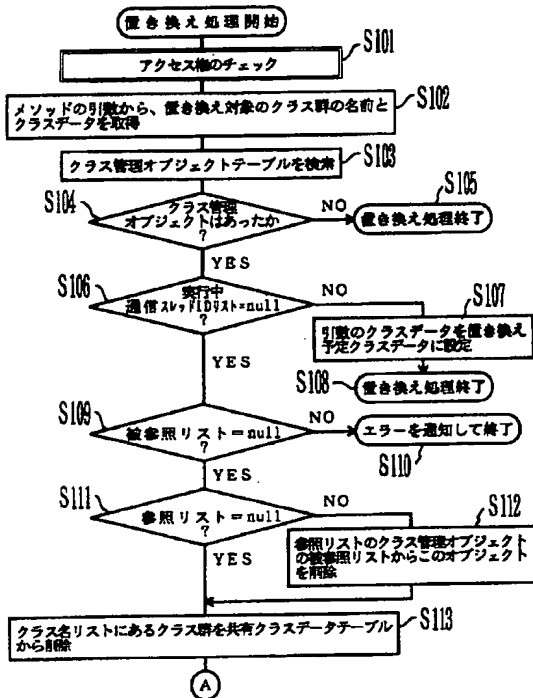
【図23】

クラスデータ置換後のエージェントクラスオブジェクトの  
状態を示す図



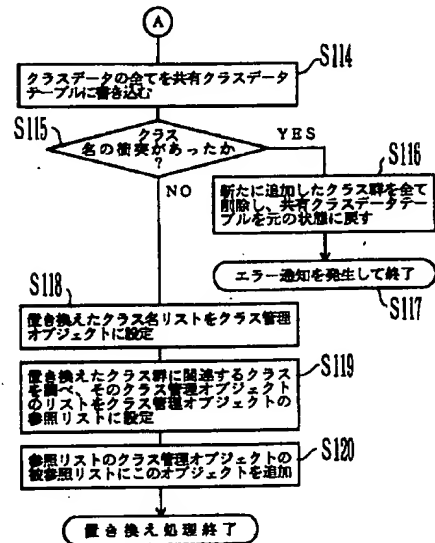
【図24】

クラスデータ置換処理を示すフローチャート

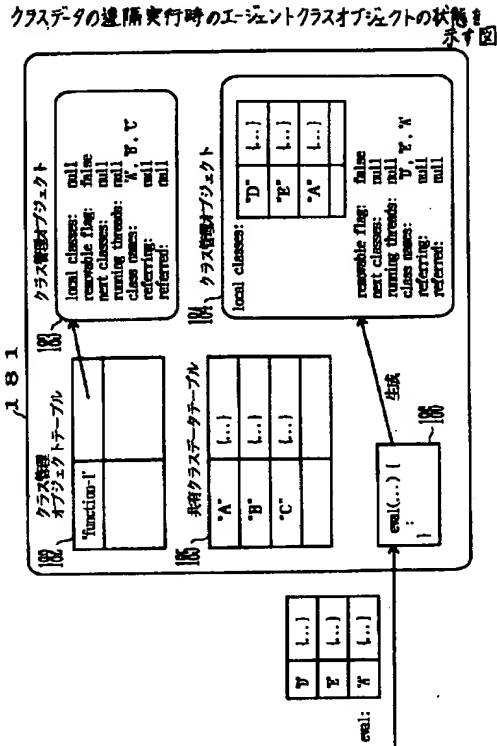


【図25】

クラスデータ置換処理を示すフローチャート(続き)

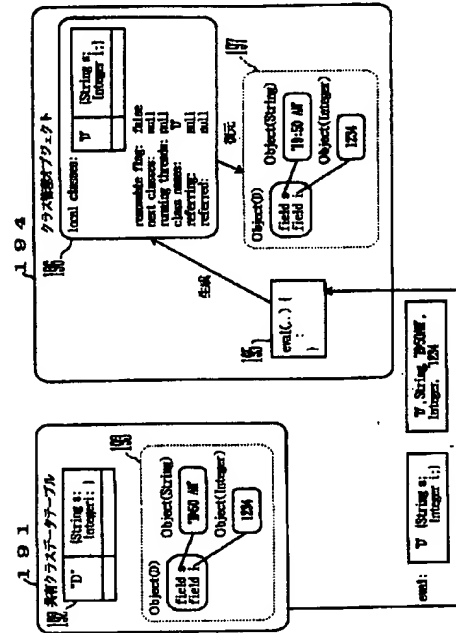


【図26】



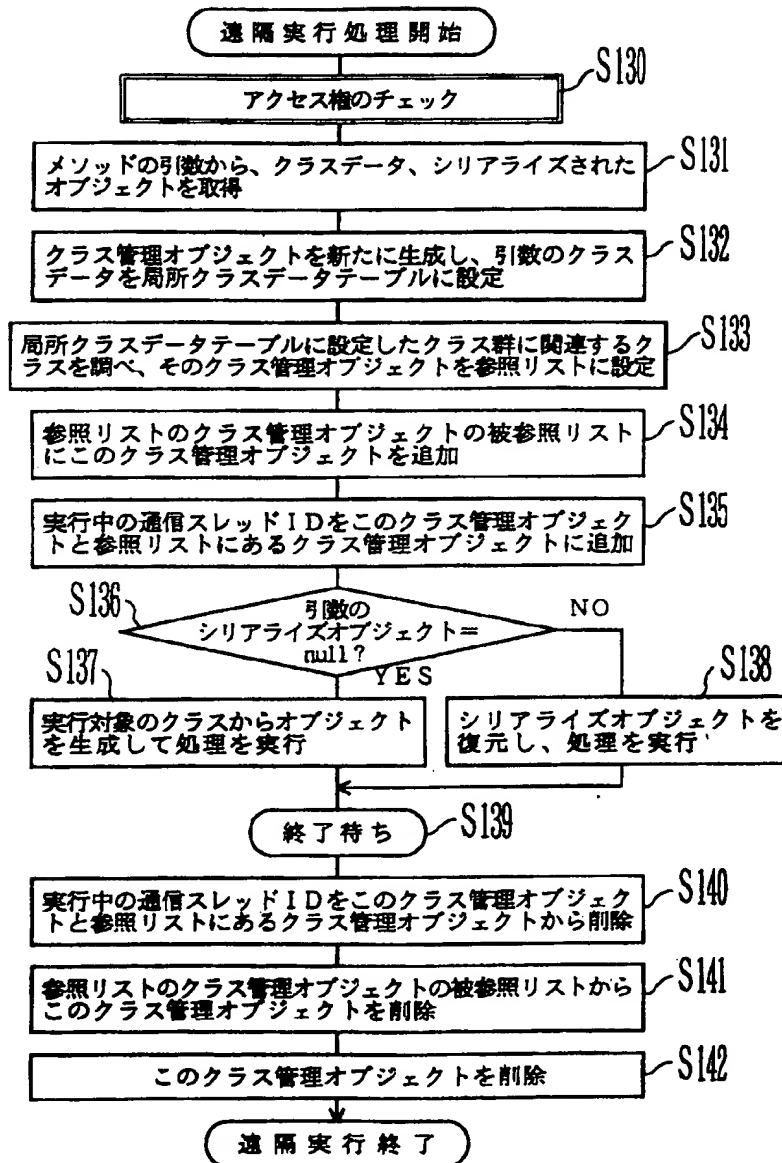
【図27】

シリアライズデータの連隔実行時の  
エージェントクラスオブジェクトの状態を示す図



【図28】

## 遠隔実行処理を示すフローチャート







【図31】

クラス検索処理を示すフローチャート

